1996

Minimum-fuel lunar transfers with engine switching and transients

Matthew Lewis Rivas
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Minimum-fuel lunar transfers with engine switching and transients

by

Matthew Lewis Rivas

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Major Professor: Bion L. Pierson

Iowa State University

Ames, Iowa

1996

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Matthew Lewis Rivas

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Historical Background

Future exploration and development of space will require the use of propulsion systems which are more efficient than any of today's conventional chemical rockets. Several alternative propulsion systems have been developed to fill this high-energy mission role. The highest efficiencies to date have been attained with electrical propulsion systems which are usually powered by a nuclear generator and are called nuclear electric propulsion systems (NEP) [1]. The analysis of NEP trajectories has been recently studied by Kleuver and Pierson [2], Hermel, et. al. [3], and Guelman [4] and all point to large fuel savings when NEP systems are used. The current use of todays electrical engines are primarily in satellite station keeping, but several small missions are planned for comet rendezvous and asteroid exploration [5]. Another propulsion system with a higher efficiency than chemical rockets, are nuclear thermal rockets (NTR) [6]. It is not widely known that President Kennedy's famous speech [7] calling for the manned Lunar missions also called for the development of a nuclear thermal rocket (NTR). This lead directly to the NERVA/KIWI programs of the late 1960's and early 1970's. This program developed and built several prototypes which provided the basic research and data needed to develop any later work. Budget restrictions and changing mission priorities stopped the program in 1972, but during the 1980's the TIMBERWIND [8] program sponsored under the Strategic Defense Initiative continued research into NTR. Also, NTR missions developed for a Lunar transportation system [9] show that not only does NTR outperform

chemical systems they provide a highly robust and competitive alternative to low-thrust or electrical systems. This is particularly true when the NTR can use a refueling method produced "locally" at a Lunar outpost as proposed by Borowski [10]. The use of NTR systems introduce periods of startup and shutdown where the rocket characteristics are not at their "rated" values and are called transients. The optimization of a trajectory is usually carried out separately from the propulsion system control, but it is possible to combine both the trajectory and the propulsion system optimization as done by Sachs and Dinkelmann [11] to possibly increase the performance and insure system limitations.

Dissertation Topics and Layout

The dissertation consists of six chapters and three appendices. The second chapter covers some of the numerical and optimization techniques used for this work. A general description is given and the specific way in which it's applied is left to later chapters. The intent is for the reader to refer to this chapter as a reference for remarks or methods mentioned later. The third chapter is for the solution of minimum-fuel lunar trajectories for a fixed thrust-coast-thrust firing sequence. A set of initial approximations is described and a dynamic boundary evaluation method introduced which greatly eases the solution of these type of trajectories. The full two-point boundary value problem is then presented. The fourth chapter describes the solution of minimum-fuel Earth-Moon trajectories with a variable thrust program. The thrust times and durations are governed by a switching function which uses information about the current state and costate of the system. The solution of this switching problem is, in general, very difficult to obtain. A related problem is solved using mixed-integer nonlinear programming to approximate the switching function. This solution and several relationships between the state and costate models are used to provide the solution of the full two-point boundary value problem. Numerical performance results for a range of flight times are presented

and their corresponding trajectories and switching structures. The fifth chapter introduces the modeling of a nuclear thermal rocket to the trajectory solution. The reactor core is modeled using the mono-energetic neutron kinetic equations and the neutronic heat-exchange process is approximated by a single-lumped heat-exchange model. The combination of reactor control and trajectory optimization is presented with numerical results for an optimal Lunar trajectory. The final chapter is the conclusions and presents a summary of the thesis as well as suggestions for further work. The first appendix details a numerical algorithm for mixed-integer nonlinear programming, the second appendix gives a series of header files describing an object class library, and the third appendix provides a derivation of the additional necessary conditions for an optimal control problem with both initial and terminal state constraints.

System Models

The propulsion systems that are used consist of two types. The first is electric propulsion and the second is NTR's. Electrical or ion engines provide very low thrust with a high performance. Ion thrusters which are in flight today have typical thrust values of 0.5 to 10 Newton with specific impulses in the $7(10)^3$ to $12(10)^3$ seconds [1]. These engines provide station keeping or attitude control and are also being used for low-power asteroid and exploration missions [5]. Future engines are expected to provide much larger thrust values and be used as the primary thrusters for a space mission. The ion engine uses electrical power to accelerate a stream of ionized gas through a series of attractors to produce thrust. An anode becomes a source of ions which "bubble" or are emitted off when they become highly charged and which are pulled by the cathode or attractors placed a short distance away. By appropriate design and shaping of the emitter and attractor, the flow of ions can be kept nearly parallel, and once exhausted the charge of the ionized stream is neutralized by emitting electrons. This is necessary because if

no neutralization is done the engine itself becomes charged and all of the propellant will simply return to it. The electrical power is supplied in a variety of ways. Solar arrays are currently being used for the smaller thrusters, but more demanding missions will require the use of nuclear reactors, hence the NEP name, in the several megawatt range for any mission to be feasible. NTR's use a nuclear core to heat the propellant which is then expelled through a nozzle providing the thrust. The propellant is usually hydrogen and provides a specific impulse of about 800 seconds in the NERVA/KIWI programs of the late seventies. The thrust is much larger than is capable by an electrical engine and the NERVA engine was rated at $10,000$ lb_f. The combination of the two engine types then allow a range of thrust values to be used. The range of thrust values and the demarcation between the two engine types is presented in Chapter 3 when numerical results are presented.

Two different dynamics models were used to simulate the spacecraft's flight path in the Earth-Moon system. The first is a patched-conic model using standard Newtonian two-body equations of motion. The second model uses the restricted circular three-body dynamics model. This simulation assumes that the Moon travels about the earth in a circular motion and the mass of the spacecraft is negligible when compared to the Earth or Moon. Another traditional aspect of this model is to transform the dynamic equations into a rotating coordinate system which rotates with a constant angular velocity equal to that of the Moon's circular speed. Thus, the Moon appears stationary in this rotating coordinate system. This model provides a high degree of accuracy for the main portion of the trajectory when the spacecraft is beyond ten Earth radii from the Earth and five Moon radii from the Moon. The portions of the trajectory close to either the Earth or Moon are essentially the same as the two-body dynamics which is expected. The nature of low-thrust trajectory calls for a large time spent near either the Earth or Moon. The three-body equations which are derived with the Earth as the center of the coordinate system are difficult to numerically integrate when the spacecraft is near the

Moon. Therefore, one set of three-body equations are derived with the Earth as the center and another set of equations with the Moon as the coordinate center. The use of which set is described in the particular problem statements and mission descriptions. The presentation of the trajectory paths in this dissertation are presented in this three-body system. The dimensions of the figures, unless otherwise stated, are in Earth radii, which is taken to be 6,387.145 km. The mission profile remains essentially constant for most of the dissertation and has an initial mass of 100,000 kg, an initial circular low-Earth orbit of 315 km, and a final circular low-Lunar orbit of 100 km.

CHAPTER 2 NUMERICAL AND COMPUTATIONAL METHODS

This chapter describes the numerical and computational methods used for this thesis. The first section details the nonlinear programming methods and mixed-integer nonlinear programming methods used. The second section describes the solution to optimal control problems (OCP) using discretization and/or collocation. The second section also describes an automatic allocation of the step-size or mesh density used by the discretization and the solution of 2PBVP by the relaxation method which is similar to discretization/collocation. The third section describes an object-orientated class library developed for the solution of optimal trajectories.

Minimization Methods

Two types of minimization methods are used: nonlinear programming (NLP) and mixed-integer nonlinear programming (MINLP). As the name implies, MINLP is basically the NLP with integer variables included in the problem formulation. The inclusion of integer parameter variables causes severe complications to efficiently compute a minimum. The difference between the two methods can be crudely summarized by describing their respective search patterns. A NLP method will use the current design space information, usually the current gradient, and constraint information to "step" through the design space to the constrained or unconstrained minimum. The method is essentially autonomous, requiring no information of previous steps or past design space parame-

ters. The MINLP strategy uses a far different approach. Instead of stepping through the space until a minimum point is reached, a MINLP method will eliminate regions of the design space which it has determined does not contain the minimum or are not allowed because of infeasibilities in the constraint hypersurface. The method will terminate with a solution when the solution region, that left after eliminating other regions of the design space, is small enough to ensure a numerically unique design vector.

The NLP problem has a general problem statement of:

$$\min_{x} \quad f(x) \tag{2.1}$$

$$\text{subject to:} \quad h(x) = 0 \tag{2.2}$$

$$g(x) \geq 0 \tag{2.3}$$

where $x \in \Re^n$, $h \in \Re^{m_e}$ equality constraint residual, and $g \in \Re^{m_i}$ inequality constraint residual vector.

The MINLP method has a general problem statement of:

$$\min_{x,y} \quad f(x,y) \tag{2.4}$$

$$\text{subject to:} \quad h(x,y) = 0 \tag{2.5}$$

$$g(x,y) \geq 0 \tag{2.6}$$

$$y \in Y \tag{2.7}$$

where x, h, and g have the same dimension as the NLP problem and where y is a p-vector within Y which is a set of integer variables. It's important to note that when finite bounds exist on the integer variables y,

$$y_L \leq y \leq y_U \tag{2.8}$$

the integer variables can be expressed as binary, i.e., 0-1 variables, denoted by z_i, by the following formula:

$$y = y_L + z_1 + 2z_2 + 4z_3 + \ldots + 2^{N-1} z_N \tag{2.9}$$

where N is the minimum number of binary variables needed and is given by

$$N = 1 + \text{INT}\left\{\frac{\log(y_U - y_L)}{\log 2}\right\} \tag{2.10}$$

where INT indicates the integer truncation of the term in the brackets. This expression correspondence may not be practical for very large bounds, but the binary form of the integer variables is particularly applicable to this work which is described in Chapter 4. The formulation of the MINLP problem then becomes

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{subject to:} \quad & h(x,y) = 0 \\ & g(x,y) \geq 0 \\ & y \in Y = \{0,1\}^p \end{aligned} \tag{2.11}$$

Equation (2.11) is the form that will be used for the algorithms and problems in this work when solving a general MINLP problem.

Nonlinear Programming

One of the most efficient modern methods for solving large-scale nonlinear programming problems is sequential quadratic programming (SQP). This method consists of determining a direction of search by solving an auxiliary quadratic programming problem (QPP) which approximates the constraints to first-order and then using an inexact line search to determine a step along that search direction. The implementation of a SQP minimization method includes other details such as design boundary constraints, termination criteria, restart and/or initial estimate feasibility verification, etc., which will not be discussed here. Many modifications to this basic idea have been suggested over time, including two which are used here:

- Higher-Order Constraint Satisfaction. A secondary QPP problem is solved which satisfies the constraint boundaries to a higher order. This secondary QPP uses

information obtained from the auxiliary QPP problem in the original SQP method to approximate the constraints to second order. The inexact line search is then along,

$$x^{i+1} = x^i + \alpha d_1 + \alpha^2 (d_2 - d_1) \tag{2.12}$$

where x^i is the current design vector, d_1 is the search direction from the auxiliary QPP, d_2 is the search direction from the secondary QPP, α is the step-size, and x^{i+1} is the next design vector. This allows a much faster rate of convergence and prevents the Maratos effect from occurring. The Maratos effect is the phenomenon that prevents superlinear convergence of a NLP algorithm near the solution, usually because of constraint satisfaction difficulties. ([12], [13], [14])

- Initial Feasible Point. It is sometimes necessary to obtain an initial feasible point before a full optimization can be carried out. This is usually the case when physical parameters need to be satisfied for the dynamic equations to be finite and/or stable. An efficient method of finding this point is to search along a direction, d, whose norm is a minimum. This search direction can be determined by solving a least distance problem of the type:

$$\min_d \quad \tfrac{1}{2} d^T d \tag{2.13}$$

$$\text{subject to:} \quad \nabla h d + h = 0 \tag{2.14}$$

$$\nabla g d + g \geq 0 \tag{2.15}$$

Since this is a linear approximation model of the nonlinear constraints, it is probable that the step-size along this search direction will have to be varied to insure a decrease in the constraint error. A typical merit function to determine a step-size is to choose a step which reduces the constraint norm of the form,

$$|c| = |h| + \sum_{i=1}^{m_i} \max(g_i, 0) \tag{2.16}$$

([15], [16])

The application of the discretization method, described later in the second section, to optimal control problems gives rise to a very large number of design variables and constraints. Because of this large number and the particular form of the gradients, an NLP program must be modified to handle sparsity issues which arise in the problem formulation. The most straight forward modification would be to rewrite the existing NLP and QPP codes to take advantage of these sparse matrices. This has the great disadvantage of "hard-wiring" the changes and preventing the use of this modified code for any other problem. Another disadvantage is the time required for the rewrite every time the original sparse matrix forms are modified. The best method of dealing with the sparsity issues is to use an object-orientated approach which is described in the last section of this chapter. The allows the original NLP and QPP code to "ignore" changes to the matrix forms since the routines simply manipulate the matrices in predefined ways. The NLP and QPP codes need never be modified after being written for matrix objects rather than primitive or built-in matrix definitions. Thus, the best of both worlds is achieved where the original code is left untouched and completely reusable to other problems and the memory requirements are reduced to the minimum required and matrix operations are implemented in the most efficient methods possible.

Mixed-Integer Nonlinear Programming Methods

The solution of mixed-integer nonlinear programming problems are typically solved using extensions of basic integer programming methods. One of the most widely used methods for solving the basic integer problem is the branch-and-bound algorithm (BB) [17]. The BB algorithm first tries to solve a relaxation of the original problem where the integer variables are allowed to vary between their lower and upper bounds. This relaxation problem is called the root node. If the root node does not result in an integer solution, then the root node is separated into two or more subproblems or nodes where each node fixes one of the integer variables at a unique value, e.g., if an integer variable is to lie

between 0 and 5 then six subproblems are formed where the first subproblem has an integer value of 0, the second has an integer value of 1, etc. One of the subproblems is solved and if the solution is integral then this branch of the solution is considered to be bounded and the method goes to the next node. If the subproblem solution is not integral, then the subproblem is split into another group of subproblems or nodes in the same manner as the root node. A list of the unsolved subproblems of any level is called a candidate list and is used to determine if all of the branches for the problem have been investigated. The solution of the original problem is the solution of the subproblem from the candidate list which provides the smallest performance index for an integral solution. The inclusion of real variables alters the algorithm but not the basic idea of the BB method. A mixed-integer BB algorithm has been proposed by Borchers and Mitchell [18] and a description of the algorithm follows:

1. Initialize data

2. Put the NLP relaxation of the problem into the branch and bound tree as the root node.

3. While there are unexamined subproblems on the candidate list

   (a) Pick a subproblem from the candidate list. If all subproblems on the list have a lower bound that is higher than the objective value of a known integer solution, then stop. The optimal integer solution is simply the best integer solution that has been found up to this point.

   (b) Repeat the following steps until one of the conditions is satisfied

      i. Take a step of a NLP algorithm

      ii. If the subproblem is infeasible, then drop it from consideration. Go back to step 3a.

iii. Compute a lower bound on the optimal objective value of the current subproblem. If this lower bound is greater than the objective value of a known integer solution, then eliminate this subproblem from consideration and return to step 3a.

iv. If the solution is optimal and the zero-one variables are all either zero or one, then we have found an integer solution. Record the value of this solution and go back to step 3a.

v. If the solution is optimal and a zero-one variable is fractional, then split the current subproblem into two new subproblems with the fractional variable fixed alternately at zero and one. Go back to step 3a.

vi. If the early branching heuristics described indicate that a zero-one variable might be fractional at optimality, then split the current subproblem into two new subproblems, and go back to step 3a.

This algorithm is very close to the classic integer BB algorithm. A depth-first with backtracking sorting algorithm is used to determine the choice of subproblem to investigate in step 3a. This sorting method chooses the next candidate subproblem as one of the child nodes of the current one. Backtracking occurs when a current branch has been completely enumerated or determined to be infeasible and the method returns to the last parent node with a child node unexamined. Other sorting methods are available but this one works well for my work. Step 3(b)iii requires a lower bound which can be determined from a dual problem of the relaxation and takes the form:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \quad +\lambda^T g(x,y) \\ \text{subject to:} \quad & h(x,y) \quad = 0 \end{aligned} \tag{2.17}$$

where y is allowed to vary between 0 and 1 and λ are the eigenvalues returned by the NLP algorithm step of the current iteration. The final comment on the algorithm concerns

step 3(b)v and the choice of which fractional variable to choose as the new branch. Although very sophisticated heuristics have been developed for this selection [17], I use the simple form suggested by Borchers and Mitchell [18] and choose that integer variable closest to 0.5.

The BB algorithm is straight forward to implement and provides reasonable performance for most problems. However, early numerical tests with my work indicated the need for a more robust algorithm which would be able to handle constraints which are extremely nonlinear. A literature search provided several algorithms but the Generalized Cross Decomposition (GCD) developed by Holmberg [19] and extended by Floudas [20] was chosen because of its performance tests and the recommendation of the excellent method survey of Grossmann [21]. A complete algorithm of the GCD method is given in Appendix A along with a flow diagram and description of the convergence tests. The basic idea is to use specialized subproblems to provide upper and lower bounds and constraint cuts to the original problem. The solution is found when the bounds have converged or the constraints have eliminated enough of the design space so as to provide a unique solution. The subproblems include the solution of additional simple MINLP problems. These subproblems are solved using the BB algorithm described above. A simple way of viewing the relationship between the GCD and BB methods is to view the GCD algorithm as the controller which determines what types of problems need to be solved and in what order and the BB algorithm as the worker which solves a particular problem but is unaware of the overall strategy. The GCD method proves to be robust and effective in handling the nonlinearities of the constraints for the work in this dissertation. It also provides a "near-global" solution where the theory does not explicitly assure a global solution, but numerical experience indicates that a global solution is almost always found. The primary disadvantage of the GCD algorithm is the relatively slow solution times when compared to other MINLP methods, but this is offset by the GCD's near-global search attribute.

Approximate Optimal Solution via Discretization

The general problem statement considered for an optimal control problem is: Find the control vector $u(t)$ which minimizes,

$$J = \phi\left[r\left(t_f\right), t_f\right] + \int_{t_o}^{t_f} L\left(r, u, t\right) dt \tag{2.18}$$

subject to:

$$\dot{r} = f\left(r, u, t\right) \tag{2.19}$$

and satisfying

$$\psi_o\left[r\left(t_o\right)\right] = 0 \tag{2.20}$$

$$\psi_f\left[r\left(t_f\right)\right] = 0 \tag{2.21}$$

where ψ_o specifies n_o initial state conditions and ψ_f specifies n_f final state conditions and t_f is fixed. The number of states is denoted by n_v and the number of controls by n_u. The system is discretized over time using n_s stages where each discrete point represents the state values, $r\left(t_j\right)$, at a specified time, t_j. This discretization requires the knowledge of $n_s + 1$ points which includes the boundary conditions. The number of points required is completely problem dependent and may range from twenty to several thousand. The time points have the characteristic of:

$$t_o < t_1 < t_2 \ldots < t_f = t_{n_s} \tag{2.22}$$

Satisfaction of the differential equations is accomplished by enforcing a defect equation of the general form:

$$\zeta_j = r_j - \hat{r}_j \tag{2.23}$$

where $\hat{r}_j$ is the resulting state vector from some integration formula. One type of useful integration formula is the trapezoid rule:

$$\hat{r}_j = r_{j-1} + \frac{\kappa_j}{2}\left[f_j + f_{j-1}\right] \tag{2.24}$$

where $f_j \equiv f(r(t_j), u(t_j), t_j)$. Another integration formula is the Hermite-Simpson interpolation formula:

$$\hat{r}_j = r_{j-1} + \frac{\kappa_j}{6}\left[f_j + 4\bar{f}_j + f_{j-1}\right] \tag{2.25}$$

where

$$\bar{f}_j = h(\bar{r}_j, \bar{u}_j, \bar{t}) \tag{2.26}$$

$$\bar{r}_j = \frac{1}{2}[r_{j-1} + r_j] + \frac{\kappa_j}{8}[f_{j-1} - f_j] \tag{2.27}$$

$$\bar{t} = \frac{1}{2}(t_j + t_{j-1}) \tag{2.28}$$

where the control values with the bar are provided by the design or parameter vector and indicate mid-point values. Several defect formulas are typically used for the solution of a given problem. The trapezoid defect has the advantage of robustness but also has low accuracy, while the Hermite-Simpson defect has high accuracy but requires far more computational expense. An effective approach uses the trapezoid defect for the initial solutions and then refines these solutions using the Hermite-Simpson formulas.

The design vector for the NLP algorithm then takes the form,

$$x = [\, r_o,\ u_o,\ r_1,\ u_1,\ \ldots,\ r_{n_s-1},\ u_{n_s-1},\ r_f,\ u_f \,]^T \tag{2.29}$$

for the trapezoid defects and the design vector for the Hermite-Simpson defect formula is,

$$x = \left[\, r_o,\ u_o,\ r_1,\ u_1,\ \bar{u}_1,\ \ldots,\ r_{n_s-1},\ u_{n_s-1},\ \bar{u}_{n_s-1},\ r_f,\ u_f,\ \bar{u}'_f \,\right]^T \tag{2.30}$$

The constraints for the NLP problem is the same for any defect scheme and can be written as:

$$c(x) = [\, \psi_o,\ \zeta_1,\ \zeta_2,\ \zeta_3,\ \ldots,\ \zeta_f,\ \psi_f \,]^T = 0 \tag{2.31}$$

Therefore, the individual points of the trajectory will satisfy the differential equations when the constraints are satisfied.

The form of the NLP problem resulting from the discretization method can now be completely described. The objective function for the NLP includes the scalar function ϕ of Equation (2.18) and a numerical approximation to the integral term, L. This approximation uses the design vector and some numerical integration algorithm to approximate the integral's value. A method which does not require a fixed time interval is preferable and Gaussian Quadrature is recommended since it does not require a fixed interval and a reasonable accuracy can be attained with a minimum manipulation of the defect data already obtained in the constraint evaluation. The constraints of the NLP are exactly Equation (2.31).

There are many other discretization schemes which can be used, particularly a Runge-Kutta method, but anything more complicated than the Hermite-Simpson method becomes prohibitively expensive when calculating the constraint vector. Also, numerical experience has shown that other schemes are not as stable as either the trapezoid or Hermite-Simpson method. The form of the constraint vector allows for analytic constraint gradient information. This information can be utilized for efficiency by the NLP solver. The main characteristic to note is that any defect vector ζ_j is only dependent upon r_j, r_{j-1}, u_j, and u_{j-1} so any partial derivative for any other $i \neq j$ or $i \neq j-1$ is zero. Defining the design vector as Equation (2.29) and the constraint vector as Equation (2.31) produces the constraint gradient:

$$\nabla c = \begin{bmatrix} \frac{\partial \psi_o}{\partial r_o} & & & & & & & & & \\ \frac{\partial \zeta_1}{\partial r_o} & \frac{\partial \zeta_1}{\partial u_o} & \frac{\partial \zeta_1}{\partial r_1} & \frac{\partial \zeta_1}{\partial u_1} & & & & & & \\ & & \frac{\partial \zeta_2}{\partial r_1} & \frac{\partial \zeta_2}{\partial u_1} & \frac{\partial \zeta_2}{\partial r_2} & \frac{\partial \zeta_2}{\partial u_2} & & & & \\ & & & & \frac{\partial \zeta_3}{\partial r_2} & \frac{\partial \zeta_3}{\partial u_2} & \frac{\partial \zeta_3}{\partial r_3} & \frac{\partial \zeta_3}{\partial u_3} & & \\ & & & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & & & \frac{\partial \zeta_{n_s}}{\partial r_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial u_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial r_{n_s}} & \frac{\partial \zeta_{n_s}}{\partial u_{n_s}} \\ & & & & & & & & \frac{\partial \psi_f}{\partial r_{n_s}} & \end{bmatrix} \tag{2.32}$$

where blank space indicates zeros and,

$$\frac{\partial \zeta_i}{\partial r_i} = \mathrm{I} - \frac{\kappa_\mathrm{i}}{2}\frac{\partial \mathrm{f_i}}{\partial \mathrm{r_i}} \tag{2.33}$$

$$\frac{\partial \zeta_i}{\partial u_i} = -\frac{\kappa_i}{2}\frac{\partial f_i}{\partial u_i} \tag{2.34}$$

$$\frac{\partial \zeta_i}{\partial r_{i-1}} = -\mathrm{I} - \frac{\kappa_\mathrm{i}}{2}\frac{\partial \mathrm{f_{i-1}}}{\partial \mathrm{r_{i-1}}} \tag{2.35}$$

$$\frac{\partial \zeta_i}{\partial u_{i-1}} = -\frac{\kappa_i}{2}\frac{\partial f_{i-1}}{\partial u_{i-1}} \tag{2.36}$$

and the partial with respect to an indexed array is just the Jacobian of f evaluated at that array, i.e.,

$$\frac{\partial f_i}{\partial r_i} = \left.\frac{\partial f}{\partial r}\right|_{r=r_i} \tag{2.37}$$

$$\frac{\partial f_i}{\partial u_i} = \left.\frac{\partial f}{\partial u}\right|_{u=u_i} \tag{2.38}$$

The total dimensions of Equation (2.32) is

$$n_{size} = (n_o + n_s n_v + n_f) \times ((n_v + n_u)(n_s + 1)) \tag{2.39}$$

but most of these elements are zero. The number of elements actually used is only:

$$n_{used} = n_o n_v + 2(n_v + n_u) n_v n_s + n_f n_v \tag{2.40}$$

An example will quickly illustrate the sparseness of Equation (2.32). Assume the problem consists of five state variables ($n_v = 5$), one control ($n_u = 1$), all the initial states are known ($n_o = n_v = 5$), all of the final states are free ($n_f = 0$), and the time interval is split into 100 stages ($n_s = 100$). The number of stages is dependent upon the dynamic equations, but 100 is not unusual and for many problems the number of stages is much greater. The size of the trapezoid constraint gradient matrix then is $n_{size} = 306,030$, but the number of elements which are non-zero is $n_{used} = 6025$ or 1.97 % of the total.

The Hermite-Simpson defect scheme has a very similar structure. Defining the design vector as Equation (2.30) and the constraint vector as Equation (2.31) produces the

constraint gradient:

$$\nabla c = \begin{bmatrix}
\frac{\partial \psi_o}{\partial r_o} & & & & & & & & & \\
\frac{\partial \zeta_1}{\partial r_o} & \frac{\partial \zeta_1}{\partial u_o} & \frac{\partial \zeta_1}{\partial r_1} & \frac{\partial \zeta_1}{\partial u_1} & \frac{\partial \zeta_1}{\partial \bar{u}_1} & & & & & \\
 & & \frac{\partial \zeta_2}{\partial r_1} & \frac{\partial \zeta_2}{\partial u_1} & \frac{\partial \zeta_2}{\partial \bar{u}_1} & \frac{\partial \zeta_2}{\partial r_2} & \frac{\partial \zeta_2}{\partial u_2} & \frac{\partial \zeta_2}{\partial \bar{u}_2} & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & & & \frac{\partial \zeta_{n_s}}{\partial r_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial u_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial \bar{u}_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial r_{n_s}} & \frac{\partial \zeta_{n_s}}{\partial u_{n_s}} & \frac{\partial \zeta_{n_s}}{\partial \bar{u}_{n_s}} \\
 & & & & & & & \frac{\partial \psi_f}{\partial r_{n_s}} & &
\end{bmatrix} \tag{2.41}$$

The defect Jacobians are

$$\frac{\partial \zeta_i}{\partial r_i} = \mathrm{I} - \frac{\kappa_i}{6}\left[\frac{\partial \mathrm{f_i}}{\partial \mathrm{r_i}} + 4\frac{\partial \bar{\mathrm{f}}_\mathrm{i}}{\partial \mathrm{r_i}}\right] \tag{2.42}$$

$$\frac{\partial \zeta_i}{\partial u_i} = -\frac{\kappa_i}{6}\left[\frac{\partial f_i}{\partial u_i} + 4\frac{\partial \bar{f}_i}{\partial u_i}\right] \tag{2.43}$$

$$\frac{\partial \zeta_i}{\partial \bar{u}_i} = -\frac{2}{3}\kappa_i \frac{\partial \bar{f}_i}{\partial \bar{u}_i} \tag{2.44}$$

$$\frac{\partial \zeta_i}{\partial r_{i-1}} = -\mathrm{I} - \frac{\kappa_i}{6}\left[4\frac{\partial \bar{\mathrm{f}}_\mathrm{i}}{\partial \mathrm{r_{i-1}}} + \frac{\partial \mathrm{f_{i-1}}}{\partial \mathrm{r_{i-1}}}\right] \tag{2.45}$$

$$\frac{\partial \zeta_i}{\partial u_{i-1}} = -\frac{\kappa_i}{6}\left[4\frac{\partial \bar{f}_i}{\partial u_{i-1}} + \frac{\partial f_{i-1}}{\partial u_{i-1}}\right] \tag{2.46}$$

$$\frac{\partial \zeta_i}{\partial \bar{u}_{i-1}} = -\frac{2}{3}\kappa_i \frac{\partial \bar{f}_i}{\partial \bar{u}_{i-1}} = 0 \tag{2.47}$$

where

$$\frac{\partial \bar{f}_i}{\partial r_i} = \left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial \bar{r}_i}{\partial r_i}\right) = \frac{\partial \bar{f}_i}{\partial \bar{r}_i}\left[\frac{1}{2}\mathrm{I} - \frac{\kappa_\mathrm{i}}{8}\frac{\partial \mathrm{f_i}}{\partial \mathrm{r_i}}\right] \tag{2.48}$$

$$\frac{\partial \bar{f}_i}{\partial u_i} = \left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial \bar{r}_i}{\partial u_i}\right) = -\frac{\kappa_i}{8}\left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial f_i}{\partial u_i}\right) \tag{2.49}$$

$$\frac{\partial \bar{f}_i}{\partial r_{i-1}} = \left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial \bar{r}_i}{\partial r_{i-1}}\right) = \frac{\partial \bar{f}_i}{\partial \bar{r}_i}\left[\frac{1}{2}\mathrm{I} + \frac{\kappa_\mathrm{i}}{8}\frac{\partial \mathrm{f_{i-1}}}{\partial \mathrm{r_{i-1}}}\right] \tag{2.50}$$

$$\frac{\partial \bar{f}_i}{\partial u_{i-1}} = \left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial \bar{r}_i}{\partial u_{i-1}}\right) = \frac{\kappa_i}{8}\left(\frac{\partial \bar{f}_i}{\partial \bar{r}_i}\right)\left(\frac{\partial f_{i-1}}{\partial u_{i-1}}\right) \tag{2.51}$$

The total size of Equation (2.41) is

$$n_{size} = (n_o + n_s n_v + n_f)\left((n_v + n_u)(n_v + n_u)(n_s + 1) + n_s n_u\right) \tag{2.52}$$

and the number of elements actually used is

$$n_{used} = n_o n_v + 2\left(n_v + 2n_u\right) n_v n_s - n_v n_u + n_f n_v \tag{2.53}$$

Using the previous example, the number of elements is $n_{size} = 356,530$ and only $n_{used} = 7020$ are non-zero or 1.97 % of the total available.

There are two basic methods for evaluating the constraint gradients in an efficient manner. The first and most straight forward is to use Equations (2.32) and (2.41) to directly evaluate the gradient. This has the advantage of being an exact value with a corresponding improvement in satisfying constraints and other accuracy improvements. The disadvantage is the reduction in speed if the dynamic equations are expensive. The second method is to use finite-difference approximations on those portions of the constraint gradient which are known to be non-zero. The primary advantage is speed and the disadvantage is accuracy.

Automatic Allocation of the Grid Points

The discretization method's ability to find a solution greatly depends upon the number of grid points used and where those points are placed in time. Increasing the number of points usually increases the overall accuracy, but this does not insure a sufficient accuracy in any specific portion of the trajectory. One method of making sure the integration accuracy is sufficient for all portions of the trajectory is given by Betts and Huffman [15]. They define a relative error for an interval j by

$$\epsilon_j = \max_{i=1,\ldots,n_v} \frac{\left|\hat{r}^i_{j+1} - r^i_{j+1}\right|}{\left|r^i_{j+1}\right| + 1} \tag{2.54}$$

where $\hat{r}$ is the predicted value of the state obtained from

$$\hat{r}_{j+1} - r_j = \int_{t_k}^{t_k+1} f\left(t\right) dt \tag{2.55}$$

The integral is evaluated using a numerical procedure, e.g., Simpson's rule. The number of grid points is then determined by:

1. Distribute the Error Equally. Form the ratio $\bar{z}_j$ of the relative error ϵ_j to the mean of the relative error over all intervals. If $\bar{z}_j > 4$, then add an extra grid point to the interval. If $\bar{z}_j < 1/10$, then remove a grid point.

2. Reduce the Relative Error Magnitude. Form the ratio z_j of the relative error ϵ_j to the targeted error level, which is usually a tenth of the user input relative error tolerance. If $z_j > 4$, add two new grid points to the interval. If $1/2 \leq z_j \leq 4$, add one grid point to the interval.

This method proves to be very effective. However, it does require multiple solutions in order to update the grid.

Another method to allocate the grid points [22] has the advantage of dynamically changing the mesh points as the minimization proceeds. The first step is to re-parameterize the time in terms of an auxiliary variable, q. This is just the coordinate corresponding to the mesh points themselves, i.e. $q = 1$ at $j = 1$ etc. The next step is to change the independent variable in the ODE's from t to q,

$$\frac{dr}{dt} = f \Rightarrow \frac{dr}{dq} = f\frac{dt}{dq} \tag{2.56}$$

In terms of q, the trapezoid defect 2.24 can be written as

$$\hat{r}_j = r_{j-1} + \frac{1}{2}\left[\left(f\frac{dt}{dq}\right)_j + \left(f\frac{dt}{dq}\right)_{j-1}\right] \tag{2.57}$$

The independent variable parameter κ is one for the difference formula because, by its definition, $q_j - q_{j-1} = 1 \forall j$. Note that dt/dq should accompany f wherever it appears, so the Hermite-Simpson defect formula can also be easily written. The transformation between t and q depends only on dt/dq. Its reciprocal, dq/dt, is proportional to the density of mesh points, but the proportionality constant which will vary the variable q between 1 and n_s is not known. A second parameterization, $Q(q)$, is introduced where Q is a new independent variable. The relation between Q and q is taken to be linear.

i.e.,

$$\frac{dQ(t)}{dq} = \xi \quad \frac{d\xi}{dq} = 0 \tag{2.58}$$

where ξ is a new intermediate variable. The final step is to create a desired mesh-density function,

$$\gamma(t) = \frac{dQ}{dt} = \frac{dQ}{dq}\frac{dq}{dt} \tag{2.59}$$

where γ is chosen by the user. Written in terms of q,

$$\frac{dt}{dq} = \frac{\xi}{\gamma(t)} \tag{2.60}$$

One requirement on γ is for it to be positive definite, otherwise the denominator of Equation (2.60) can be zero. Automated spacing then can be implemented by adding Equations (2.58) and (2.60). Consider the original 2PBVP,

$$\frac{dx}{dt} = f(x,t) \tag{2.61}$$

where $x \in \Re^n$ and n_o initial conditions and n_f final conditions are known so that $n = n_o + n_f$. The new system with automated grid spacing is

$$\frac{dx}{dq} = \frac{\xi}{\gamma(t)} f(x,t) \tag{2.62}$$

$$\frac{dQ}{dq} = \xi \tag{2.63}$$

$$\frac{d\xi}{dq} = 0 \tag{2.64}$$

$$\frac{dt}{dq} = \frac{\xi}{\gamma(t)} \tag{2.65}$$

Three new boundary conditions are required because of the three new equations. Two of these conditions are known because of the last equation where both the initial and final times are specified. The final boundary condition comes from specifying $Q = 0$ at the initial time which is required for q to vary between 1 and n_s. Finally, $\gamma(t)$ must be specified and is chosen by the user. A simple example would be

$$\gamma = \frac{1}{\Delta} + \frac{|dx/dt|}{\delta |x|} \tag{2.66}$$

where the first term spaces the points evenly and the second term attempts to place the points where the magnitude of the rate of change of the ODE's is greatest. The two constants, Δ and δ, specify how "attracted" a point is to a given distribution over any other, e.g., a point is approximately attracted an amount Δ to an even distribution and an amount δ to a magnitude distribution. This form for γ and the point distribution allows the ODE variables to be used directly in the automatic allocation method. This is the primary reason for the form of the time transformation, Equation (2.60).

2PBVP Solution by Relaxation

The solution of a 2PBVP can be accomplished by the relaxation technique which is very similar to the discretization method discussed in this section. This solution method is used in Chapter 4 to solve the switching problem from the MINLP approximation. The relaxation method uses a defect formula to link a series of discretized time points together to both satisfy the boundary conditions and ODE's in the same manner as the discretization method. The difference is that no control vector is present and a discretized point must satisfy the ODE with no other influence. This eliminates the use of the Hermite-Simpson defect formula since control midpoint values are needed. Since the use of the relaxation technique in this work will be limited to solving a combined state-costate system which results from the application of optimal control theory, the ODE vector will be written as

$$y = \begin{bmatrix} r \\ \lambda \end{bmatrix} \tag{2.67}$$

where λ is a secondary vector the same dimension as r thus y is $2n_v$ in length. The right-hand side of the ODE is then

$$\dot{y} = F(r, \lambda, t) = \begin{bmatrix} \dot{r} \\ \dot{\lambda} \end{bmatrix} \tag{2.68}$$

where

$$\dot{r} = f(r, \lambda, t) \tag{2.69}$$

$$\dot{\lambda} = -\left(\frac{\partial L}{\partial r}\right)^T - \left(\frac{\partial f}{\partial r}\right)^T \lambda \tag{2.70}$$

Equation (2.69) is the same as Equation (2.19) except that it is assumed that the control u can be replaced with some combination of r and λ. Equation (2.70) is the standard formula for the costate equations with the integral L term present in the problem statement. The method of setting the defect vectors to zero and satisfying the ODE and boundary conditions can be accomplished using an NLP algorithm. The design vector becomes

$$x = [\, y_o, \;\; y_1, \; \cdots \, , \;\; y_{n_s-1}, \;\; y_f, \,]^T \tag{2.71}$$

which is of length $2n_v\,(n_s+1)$. The constraint vector is the same as Equation (2.31) with a constraint gradient of

$$\nabla c = \begin{bmatrix} \frac{\partial \psi_o}{\partial y_o} & & & & \frac{\partial \psi_o}{\partial y_{n_s}} \\ \frac{\partial \zeta_1}{\partial y_o} & \frac{\partial \zeta_1}{\partial y_1} & & & \\ & \frac{\partial \zeta_2}{\partial y_1} & \frac{\partial \zeta_2}{\partial y_2} & & \\ & & \ddots & \ddots & \\ & & & \frac{\partial \zeta_{n_s}}{\partial y_{n_s-1}} & \frac{\partial \zeta_{n_s}}{\partial y_{n_s}} \\ \frac{\partial \psi_f}{\partial y_o} & & & & \frac{\partial \psi_f}{\partial y_{n_s}} \end{bmatrix} \tag{2.72}$$

where

$$\frac{\partial \zeta_i}{\partial y_i} = \mathrm{I} - \frac{\kappa_\mathrm{i}}{2}\frac{\partial \mathrm{F_i}}{\partial \mathrm{y_i}} \tag{2.73}$$

$$\frac{\partial \zeta_i}{\partial y_{i-1}} = -\mathrm{I} - \frac{\kappa_\mathrm{i}}{2}\frac{\partial \mathrm{F_{i-1}}}{\partial \mathrm{y_{i-1}}} \tag{2.74}$$

The Jacobian of the combined system involves the second partials of the original dynamics and takes the form

$$\frac{\partial F}{\partial y} = \begin{bmatrix} f_r & f_\lambda \\ \left(-L_r - f_r^T \lambda\right)_r & -f_r^T \end{bmatrix} \tag{2.75}$$

The two diagonal terms are the Jacobian of the original system. The upper right term is the Jacobian of the original system with respect to the costate. The terms here are determined by how the control was replaced by the state and/or costates. The lower left term involves second partials of both L and f. The boundary conditions are imposed by ψ_o and ψ_f. These functions are changed from the discretization formulation by becoming dependent on both the initial and final points. This was done because the boundary conditions imposed on the costate vector often involve initial and terminal state and costate information. No difficulty is encountered introducing the new Jacobians except that the matrix is no longer banded. The dimension of the Equation (2.72) is $(2n_v n_s + n_o + n_f) \times (2n_v (n_s + 1))$ where $n_o + n_f = n_v$ is required for the solution of a 2PBVP. The gradient is square and the problem of zeroing the constraints is simply a feasibility problem. This type of problem does not require a performance value for solution, however, a simple version such as

$$\min_x c^T W c \tag{2.76}$$

where W is some positive definite matrix, can be used for NLP algorithms without a feasibility option. The relaxation solution process can now be described. An initial estimate of the vector y at each of the time points is provided to an NLP algorithm which then attempts to zero the constraint vector, Equation (2.31). A performance index may or may not be present depending on the capabilities of the NLP algorithm.

Object Class Library

The purpose of any class library is to ease the computational burden of the programmer to perform any given task. The vast majority of libraries available are for interfacing with the user in a graphic interface, i.e., a GUI. Few libraries have been developed for numerical work, and these have concentrated on linear algebra techniques. The benefits of a class library include code re-usability, efficient and easy extensions to the existing

code structure, and guaranteed memory and speed improvements versus a similar serial programming technique.

This section describes a class library that was developed for this work. The general layout of the library, descriptions of the major branches, and implementation details are given. The discussion will center on general object-orientated topics to broaden the concepts to any available computer language which supports objects. C++ is the actual language the library was written in, and corresponding header files are given in Appendix B. The number of references dealing with numerical object libraries is somewhat limited but the book by Ferraris [23] is good, especially if the reader is from a Fortran background. Other texts on object libraries in general are readily available including [24] and [25].

Object-Orientated Terminology and Concepts

The main advantage of using a class library for my work is the ability to extend existing classes into new forms for similar work. The two main ways to extend a class is by encapsulation or derivation. By definition, a class bundles together data and functions. Encapsulation is the method where part of the data bundled is another class. Derivation is the method where a new class is derived from another. This derivation allows the derived or child class to have access to all of the data and functions bundled in the base or parent class. The child class can then add new data or extend existing functions for new uses. The most important point of derivation is that if one class encapsulates another, any child of the encapsulated class can be substituted for the original. This concept will be more clearly illustrated by the examples and class structure.

Another advantage which will allow very efficient memory and speed performance increases is operator overloading and/or polymorphism. An operator is something which manipulates two objects to produce a third. The easiest example is any of the basic operators found in all languages: addition, subtraction, multiplication, division, etc.

The addition of two integers produces a third integer. All object languages allow the programmer to overload certain operators or to define them for objects which the programmer has created. I overload these operators for a linear algebra matrix class. There are several public domain linear algebra packages which essentially do the same thing. My implementation will take advantage of the specific matrix forms generated by the discretization method discussed in the previous section.

The general class structure has two main forms: the big tree-small forest or small tree-big forest. A class structure is the way the individual objects are arranged and related to each other, i.e., which objects share information, the derivation chains, and allowable and/or required extensions to current objects. The big tree-small forest approach is illustrated in Figure 2.1. The main idea is to have all objects derived from a single

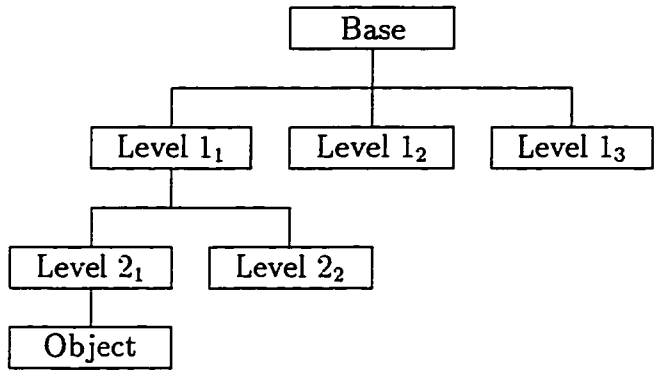

Figure 2.1 Class Library Structure Example 1

base. This base would then encompass as many functions and data as possible. The first and second levels are then used to include new functions if necessary and to implement a specific task. The object at the end is what the user would need to perform his task. The advantage of this type of class structure is the simplicity and the ease by which modifications to the base class do not effect the class structure. The primary disadvantage is the need to include unnecessary information in the final object. The object in the figure might only require a portion of the data the base class bundles,

but must include it all for the small part it needs. This can lead to wasted memory and speed as well as a large amount of bookkeeping. The name describes how the class appears to be a large tree with many branches all connected to the trunk.

The little tree-big forest model takes a very different approach to the class structure and is show in Figure 2.2. Several bases are defined where each will implement a specific

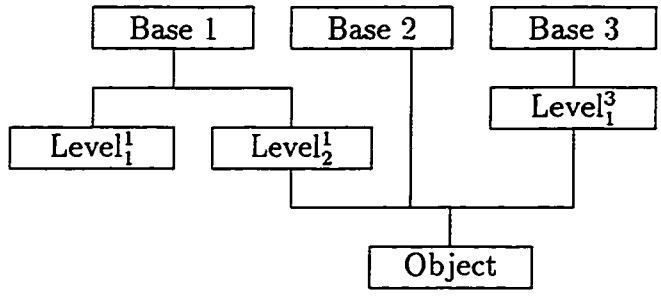

Figure 2.2 Class Library Structure Example 2

task. Same derivation is done to implement specific modifications, but the user's object is derived from multiple sources. This eliminates the waste of including unnecessary data and functions, but introduces the possibility of incompatibilities between the individual bases. Another advantage of this model is the ability to include other class libraries developed elsewhere in the final application. This allows a GUI, which can be platform specific, to coexist with this class structure. The big tree-little forest model can only include other libraries with great difficulty. The name comes from the many individual "trees" which are combined into the useful object.

Some quick examples will provide a better understanding of the class structures described above. Several numerical algorithms require the user to provide a function which the algorithm then uses to solve the user's specific question. This function is sometimes identical over several algorithms, e.g., numerical integration, interpolation,

and root-finding all require a function of the form:

$$f = f(x) \tag{2.77}$$

or given an argument return the function value at that argument. A big tree-little forest model can then define such a function in its base. The separate levels derived from this base can then implement the algorithms themselves. This example shows how this model is more easily applied to simple algorithms which only require a small user input. Many problems require the inclusion of several different numerical techniques to solve the problem. An optimal trajectory usually requires an NLP algorithm, ordinary differential equation solution method, and other supporting routines such as interpolation, root-finding, and linear algebra packages. Rather than attempting to define one base which can encompass all of these routines, several bases can be defined and the useful object can be derived from any or all of these bases. This example shows the usefulness of the little tree-big forest model for handling a wide range of very different algorithms in an efficient way.

Library Description

The class structure developed for this work uses a combination of both models mentioned in the previous section, but primarily depends on the little tree-big forest model. Figure 2.3 shows the class layout. All lines ending in an arrow indicate direct derivation while any line without an arrow, such as that linking the DYNAMICS class to the OPTIMAL CONTROL class, indicates an encapsulation. A dashed box indicates a pure virtual base class. This means that no instance of this class can be created and only classes derived from it are allowed. This prevents the user from trying to use a class where the user must override some function to provide problem specific behavior. The numerical integration algorithm mentioned above is an example.

Three different sections are defined as the dynamic models, the linear algebra matrix

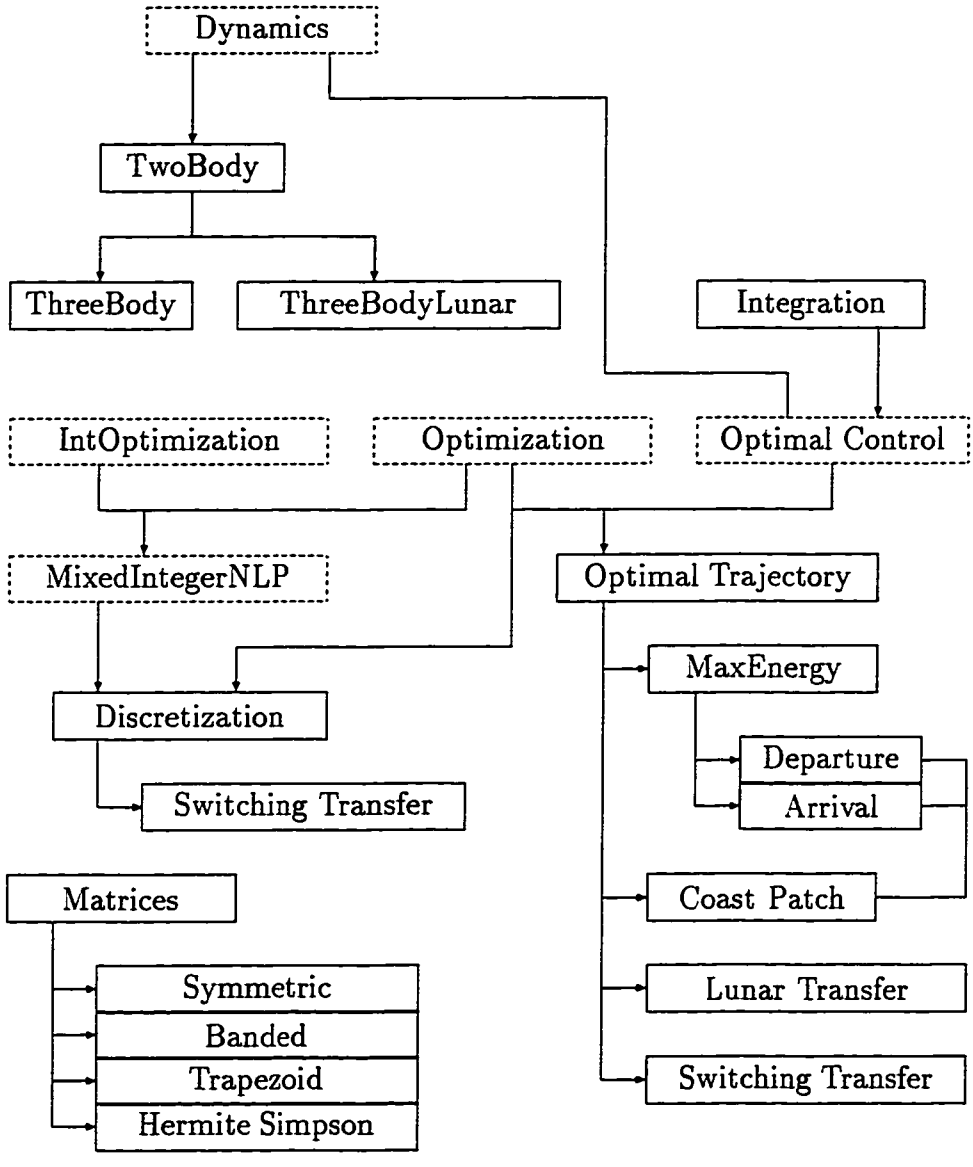

Figure 2.3 Class Library Diagram

classes, and the optimization and optimal control methods. The optimization and optimal control classes encapsulate both the dynamics models and linear algebra classes. The dynamic models are encapsulated in a specific way, as shown, while the matrix class is used whenever necessary internally by all of the classes.

The matrix class is the simplest and easiest to describe. It has a single base which defines the basic behavior of a matrix object, specifically how the memory is stored and how operators are handled. The operators which are overloaded include addition, subtraction, multiplication (by both another matrix and a scalar), and several numerical algorithms to aid in finding inverses, solutions to a set of equations, etc. This allows the matrix objects to be operated on as if they were scalar variables. The base class implements these operations in the standard matrix method, i.e.,

$$c_{ij} = a_{ij} + b_{ij} \tag{2.78}$$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \tag{2.79}$$

where c_{ij} is the element of the resulting matrix C in the ith row and jth column. The derived classes implement specific matrix behavior, e.g., the SYMMETRIC class changes the memory allocation to only store the required $n\,(n+1)$ elements and when multiplied by another SYMMETRIC matrix object will return a SYMMETRIC matrix object. This multiplication is also done without performing matrix multiplications that are not required, i.e., the lower triangle portion of the matrix. No special implementation details are required from the user and the advantages of using SYMMETRIC matrices are carried out regardless of whether the user was aware of it happening or not. This is the guarantee that only the necessary memory is allocated and all matrix manipulations are carried out in the most efficient manner. The last three classes are a BANDED class which is useful in the NLP algorithm, and the TRAPEZOID and HERMITE SIMPSON classes which implement the specific form of the discretization matrices produced and described in the second section. They also only allocated the absolute minimum amount

of memory necessary. This is of primary importance for the discretization method since without some memory saving the method is impractical for most problems.

The dynamic model class implements the three dynamic models used in this work which include two-body Newtonian dynamics with thrusting and restricted circular three-body dynamics centered on the Earth and the Moon with and without thrusting. The single base class defines a dynamic equation subroutine which returns a derivative vector from a given state vector, control vector, and time. A Jacobian function is also defined for any dynamic model. The default numerically calculates this Jacobian, but the user can override this default for any derived classes. This is done for all three models mentioned. The OPTIMAL CONTROL class, discussed below, requires a dynamic model. This is logical since a dynamic model is necessary to define an optimal control problem.

The optimization and optimal control classes take full advantage of multiple inheritance to mix classes and provide an efficient implementation. The INTEGRATION class is the most removed from optimization and control theory and integrates a set of ordinary differential equations from some initial time to some final time using the Bulirsch-Stoer integration scheme [22]. The OPTIMAL CONTROL class implements the basic concepts of optimal control. The class will create the costate differential equations from the Jacobian of the dynamic object, even if this dynamic model is a numerical approximation. The class will also integrate the couple state and costate equations using the methods provided by the INTEGRATION class. It's a pure virtual class that requires the user to give the class the form of the optimal control. There is a default which finds this optimal control by solving the stationarity condition of the Hamiltonian by a numerical root-finding method. However, this is impractical for any but the simplest problems. The second base class is called the OPTIMIZATION class and implements a NLP algorithm to optimize some object function which is problem specific which also makes this class pure virtual. The INTOPTIMIZATION class uses the branch-and-

bound algorithm, described in the first section, to optimize an integer objective function which again is problem specific. The MIXEDINTNLP class implements the GCD algorithm mentioned also described in the first section. This algorithm requires the solution of several subproblems which may be solvable by an NLP method and/or IP method, which this class can now do because of its derivation from the two previous classes mentioned. The last general class is the OPTIMAL TRAJECTORY class. This class bridges the gap between optimal control theory (OPTIMAL CONTROL class) and the solution of these problems by an NLP algorithm (OPTIMIZATION class). Several methods are implemented, but the one most used is the terminal error function method. The method uses the unknown initial conditions from the two-point boundary value problem (usually the costates) as the design variables of the NLP algorithm. The objective function is defined as the difference from the known final conditions and the resulting final conditions after the combined system has been integrated forward in time using the routines available to OPTIMAL CONTROL and hence OPTIMAL TRAJECTORY because of its derivation. An example is given in Chapter 3 where this terminal error method is described more fully when solving the maximum-energy problem.

The rest of the classes are direct problem specific implementations which solve actual problems. The MAXENERGY class solves the general maximum-energy problem described in Chapter 3. The COAST PATCH, DEPARTURE, and ARRIVAL classes implement the dynamic boundary evaluation also described in Chapter 3. The DISCRETIZATION class solves an optimal trajectory with integer design variables for the alternative switching function structure described in Chapter 4. A complete description of each class would entail the theory and derivation given in the following chapters. Because of this and the fact that this discussion is limited to the class structure and not the implementation details, no further discussion is given here of these classes.

CHAPTER 3 LUNAR TRANSFERS

Minimum-fuel transfers between a circular low-Earth orbit (LEO) and a circular low-Lunar orbit (LLO) are described in this chapter. A description of the subproblem hierarchy and a dynamic boundary evaluation method (DBE) for efficiently calculating an initial estimate for the transfer is given. The method of finding the initial estimate was developed by Pierson and Kluever [26], and the DBE extension is described in Rivas and Pierson [27]. This initial estimate is then used to solve both a "hybrid" direct/indirect transfer and the full optimal transfer which solves the two-point boundary value problem (2PBVP). The transfers considered in this chapter are all assumed to have a thrust-coast-thrust engine firing sequence where the separate stage time durations are determined by the solution. This assumption simplifies the initial analysis and provides information about other engine firing sequences which are discussed in the next chapter.

Initial Approximations

A typical trajectory spirals out from the Earth until enough energy is obtained to begin a coast phase. This coast phase takes the spacecraft from the Earth to near the Lunar sphere of influence where the next thrust phase spirals into the desired LLO. The solution of low-thrust Lunar transfers is extremely sensitive to initial estimates of the optimal control where small errors in the initial estimate of the thrust steering angle can cause large errors in the boundary conditions. It becomes necessary to obtain an initial estimate of the trajectory in order for the general transfer to be solved. A very efficient

way of finding this initial estimate is described in this section.

The basic method solves a hierarchy of subproblems which build up to the minimum-fuel transfer. The thrusting portions of the trajectory are modeled as maximum-energy spirals. The coast portion finds a non-thrusting trajectory which combines two maximum-energy spirals and minimizes the amount of fuel used. Finally, the trajectory is fully modeled using the controls for the spirals and coast.

Maximum Energy Spirals

A maximum-energy spiral is one which uses continuous thrusting to maximize the total energy of the spacecraft at the end of the trajectory for a given time-of-flight. These trajectories are easy to solve for a range of flight times, initial radii, and various initial thrust-to-weight ratios. They also provide adequate boundary conditions for the translunar coast mentioned previously and described in the next section. The maximum-energy spiral can model both the Earth departure and Lunar capture. The Earth departure is modeled as a maximum-energy spiral since the initial conditions are known. The Lunar capture has the same form, i.e., a max-energy spiral, but is integrated backwards in time since the final conditions are known. The total energy of the spacecraft is then maximized at the beginning of the capture.

The problem statement for a maximum-energy spiral is given by:
Find the thrust direction time history $u(t)$, $t_0 \leq t \leq t_f$, which minimizes

$$J = -\left(\frac{V_r^2 + V_\theta^2}{2} - \frac{\mu}{r}\right)_{t=t_f} \tag{3.1}$$

subject to

$$\dot{r} = V_r \tag{3.2}$$

$$\dot{\theta} = \frac{V_\theta}{r} \tag{3.3}$$

$$\dot{V}_r = \frac{V_\theta^2}{r} - \frac{\mu}{r^2} + a \sin u \tag{3.4}$$

$$\dot{V}_\theta = -\frac{V_r V_\theta}{r} + a\cos u \tag{3.5}$$

where μ is the gravitational parameter for the attracting body, i.e., the Earth or Moon, a is the thrust acceleration defined by

$$a = \frac{T}{m} = \frac{T}{m_o - \beta t} \tag{3.6}$$

where T is the constant thrust, β is a constant mass flow parameter, and m_o is the initial mass of the spacecraft. The final time is fixed. The massflow is assumed constant so the mass m can be replaced by a linear equation in time. The boundary conditions for an initial circular orbit are

$$r(t_o) = r_o \tag{3.7}$$

$$\theta(t_o) = 0 \tag{3.8}$$

$$V_r(t_o) = 0 \tag{3.9}$$

$$V_\theta(t_o) = \sqrt{\frac{\mu}{r_o}} \tag{3.10}$$

Equations (3.2) - (3.5) describe a thrusting spacecraft in an inverse-square gravity field using polar coordinates. The variables are the radial position r, polar angle θ, radial velocity V_r, and tangential velocity V_θ. The thrust direction angle, u, is measured from the local horizon and is positive above the horizon and in the direction of the spacecraft motion. Note that the polar angle is not necessary to specify the trajectory since it does not appear on the right-hand side of the differential equations, but it is included here for convenience.

The Hamiltonian [28] is defined as

$$H = L + \lambda^T f = \lambda^T f \tag{3.11}$$

where $L = 0$ since no integral term is in the performance index and f represents the dynamics of the system under consideration. The costate vector is given by

$$\lambda = [\ \lambda_r,\ \lambda_\theta,\ \lambda_{V_r},\ \lambda_{V_\theta}\ ]^T \tag{3.12}$$

so the Hamiltonian becomes

$$H = \lambda_r \dot{r} + \lambda_\theta \dot{\theta} + \lambda_{V_r} \dot{V}_r + \lambda_{V_\theta} \dot{V}_\theta \tag{3.13}$$

Applying the stationarity condition yields

$$H_u = 0 \Rightarrow (f_u)^T \lambda = 0 \Rightarrow a\left(\lambda_{V_r} \cos u - \lambda_{V_\theta} \sin u\right) = 0 \tag{3.14}$$

Canceling out and rearranging gives the classic "tangent steering law":

$$\tan u^* = \frac{\lambda_{V_r}}{\lambda_{V_\theta}} \tag{3.15}$$

$$\Rightarrow \sin u = \frac{\pm\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.16}$$

$$\cos u = \frac{\pm\lambda_{V_\theta}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.17}$$

$$\tag{3.18}$$

The Legendre-Clebsch sufficient condition [28] provides the correct sign for $\sin u$ and $\cos u$:

$$0 \leq H_{uu} \tag{3.19}$$

$$\leq -a\left(\lambda_{V_r} \sin u + \lambda_{V_\theta} \cos u\right) \tag{3.20}$$

$$\leq -a\left(\frac{\lambda_{V_r}\left(\pm\lambda_{V_r}\right) + \lambda_{V_\theta}\left(\pm\lambda_{V_\theta}\right)}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}}\right) \tag{3.21}$$

Therefore, the negative sign is needed, i.e.:

$$\sin u^* = \frac{-\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.22}$$

$$\cos u^* = \frac{-\lambda_{V_\theta}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.23}$$

The optimal control, u^*, is completely defined by the velocity costate time histories. The necessary conditions for the costate system yields the costate differential equations

$$\frac{d\lambda}{dt} = -\left(\frac{\partial H}{\partial \bar{r}}\right) = -\left(\frac{\partial f}{\partial \bar{r}}\right)^T \lambda \tag{3.24}$$

where $\bar{r} = [\, r,\ \theta,\ V_r,\ V_\theta \,]$. The Jacobian of the state equations, $(\partial f/\partial \bar{r})$, is easily found to be:

$$\left(\frac{\partial f}{\partial \bar{r}}\right) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -\frac{V_\theta}{r^2} & 0 & 0 & \frac{1}{r} \\ -\frac{V_\theta^2}{r^2} + \frac{2\mu}{r^3} & 0 & 0 & \frac{2V_\theta}{r} \\ \frac{V_r V_\theta}{r^2} & 0 & -\frac{V_\theta}{r} & -\frac{V_r}{r} \end{bmatrix} \tag{3.25}$$

The costate equations are found by using the Jacobian (3.25), the costate equation definition (3.24), and the equations found from the stationarity condition (3.22) - (3.23):

$$\dot{\lambda}_r = \frac{V_\theta}{r^2}\lambda_\theta - \left(\frac{2\mu}{r^3}\lambda_\theta - \frac{V_\theta^2}{r^2}\right)\lambda_{V_r} - \frac{V_r V_\theta}{r^2}\lambda_{V_\theta} \tag{3.26}$$

$$\dot{\lambda}_\theta = 0 \tag{3.27}$$

$$\dot{\lambda}_{V_r} = -\lambda_r + \frac{V_\theta}{r}\lambda_{V_\theta} \tag{3.28}$$

$$\dot{\lambda}_{V_\theta} = -\frac{\lambda_\theta}{r} - \frac{2V_\theta}{r}\lambda_{V_r} + \frac{V_r}{r}\lambda_{V_\theta} \tag{3.29}$$

The boundary conditions for these costates are found by the transversality condition:

$$\lambda(t_f) = \frac{\partial \phi}{\partial \bar{r}} \tag{3.30}$$

which yields

$$\lambda_r(t_f) = -\frac{\mu}{r^2(t_f)} \tag{3.31}$$

$$\lambda_\theta(t_f) = 0 \tag{3.32}$$

$$\lambda_{V_r}(t_f) = -V_r(t_f) \tag{3.33}$$

$$\lambda_{V_\theta}(t_f) = -V_\theta(t_f) \tag{3.34}$$

It should be noted that the costate differential equation for the polar angle $\dot{\lambda}_\theta$ and its boundary condition show that it is zero for all time.

The 2PBVP is then defined by Equations (3.2) - (3.5) and Equations (3.26) - (3.29) with boundary conditions (3.7) - (3.10) and (3.31) - (3.34). The solution of this 2PBVP is obtained by means of a terminal error function approach and is related to the shooting

method [22]. A guess of the unknown initial costate conditions is used to numerically integrate the coupled differential equations forward in time to t_f. A measure of the error between the final values of the resulting costates and the desired values is made and initial costates are then modified so as to decrease this error measure. A NLP algorithm, such as described in Chapter 2, is used to find this modification. The design variables for the NLP are the unknown initial costate equations: λ_r, λ_{V_r}, λ_{V_θ}, and λ_m. The objective function used by the NLP for the 2PBVP is

$$F = \left(\lambda_r + \frac{\mu}{r^2(t_f)}\right)^2 \tag{3.35}$$

with the rest of the final costate boundary conditions being strictly enforced through constraints in the NLP. A minimum objective function of zero with constraint satisfaction is the solution to the 2PBVP. Any of the constraints could be used in an objective format similar to Equation (3.35) and the results would be identical.

An example of a maximum-energy spiral about the Earth is given in Figure 3.1 for a flight time of 2.4 days. The spacecraft is assumed to have an initial thrust-to-weight ratio of $3(10^{-3})$. The optimal control for this trajectory is given in Figure 3.2. The combined state/costate equations (3.2) - (3.5) and (3.26) - (3.29) are numerically integrated using a Bulirsch-Stoer extrapolation scheme [22], and the problem is solved using the terminal error function approach described above.

The maximum-energy problem for an Earth spiral can be solved for a range of flight times, and the relationships between the time-of-flight, final radius, and final velocity components are shown in Figures 3.3 and 3.4. Eight solutions are shown with a final radius between 4 and 18 Earth radii. The flight time and velocity components are plotted versus the final Earth radius. The curves are seen to be smooth and well behaved.

A Lunar spiral can also be solved for using the maximum-energy problem definition, but with the total energy being maximized at the initial time. The spacecraft is then

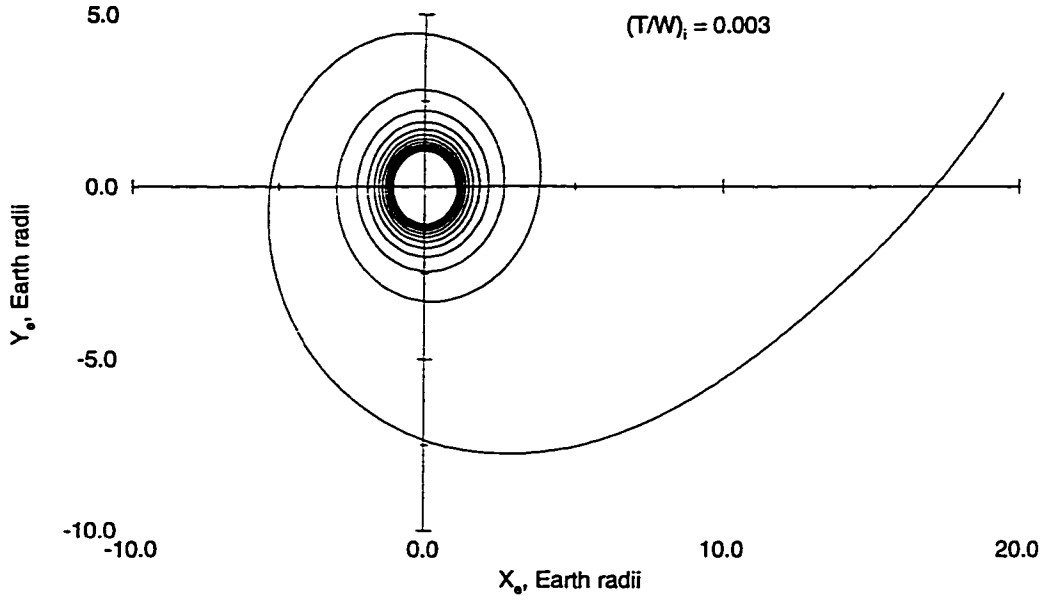

Figure 3.1 Maximum-Energy Earth Trajectory for $t_f = 2.5$ days

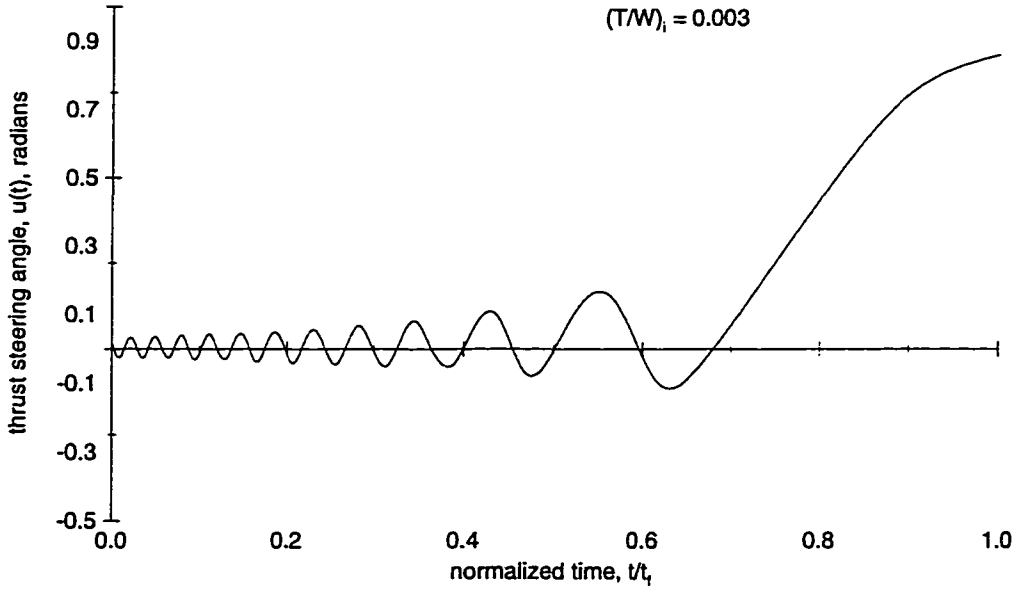

Figure 3.2 Maximum-Energy Optimal Control for Earth Trajectory

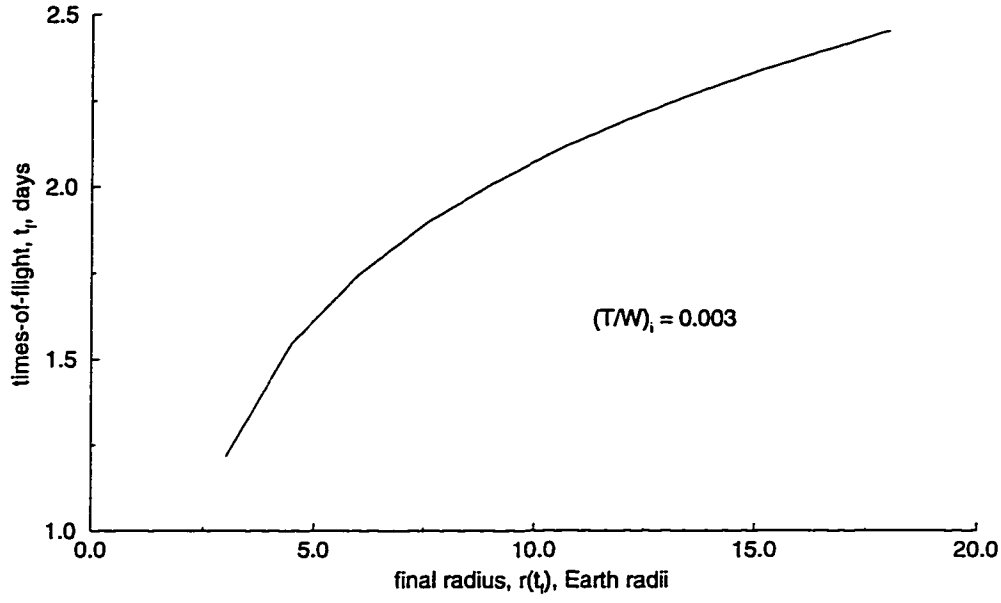

Figure 3.3 Flight Time of Earth Spiral vs. Final Radial Distance

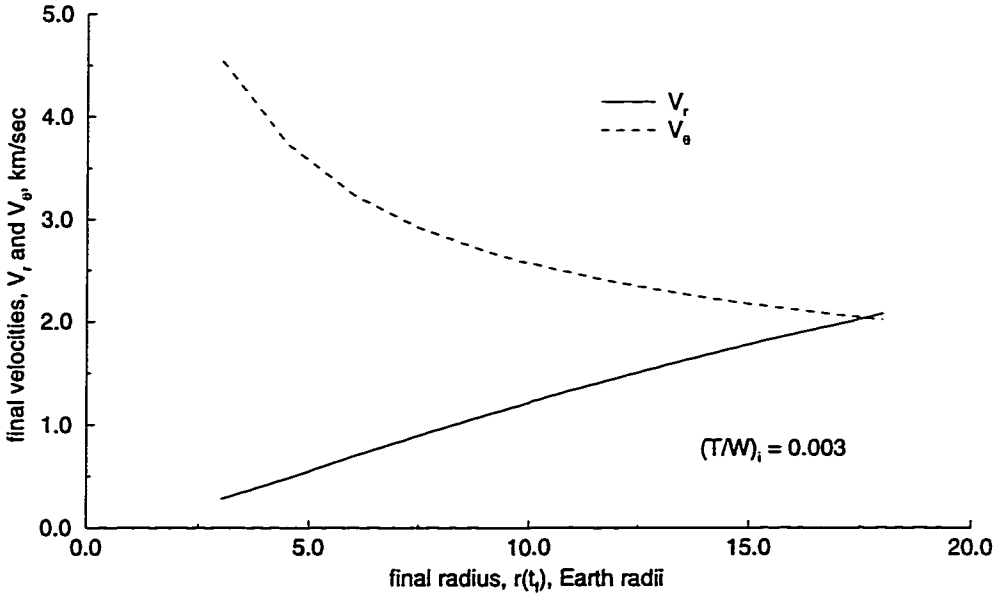

Figure 3.4 Final Velocity Components of Earth Spiral vs. Final Radial Distance

steered into the desired LLO. The equations are integrated backwards in time, since the initial LLO is known. Another parameter which must be included is the mass at the end of the trajectory, since for any given Lunar capture spiral it is not known which specific Earth escape spiral was used. Except for these differences, the problem formulation is identical to the Earth spiral. Figure 3.5 shows the Lunar spiral for a flight time of 16

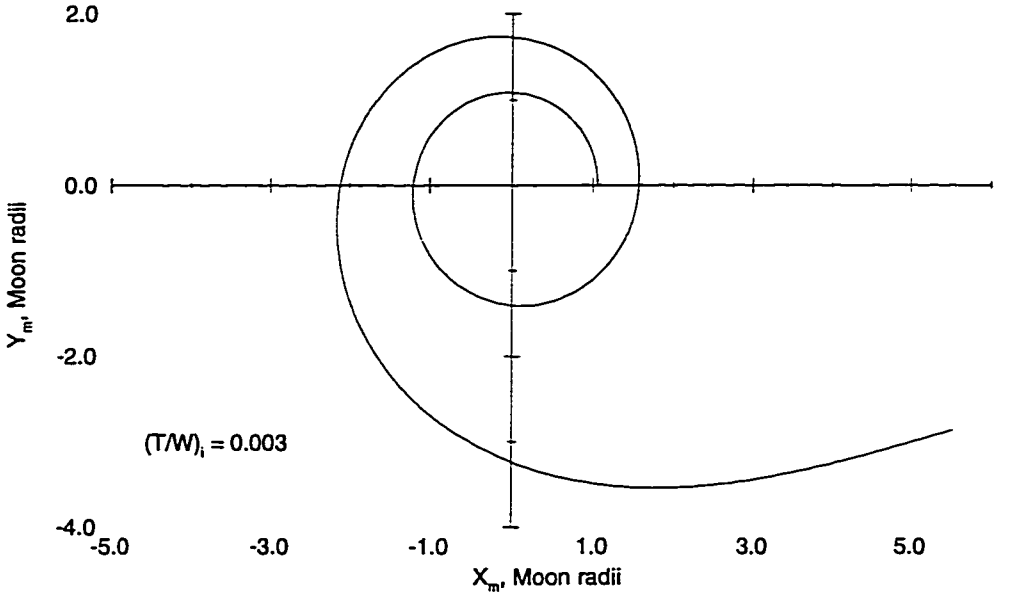

Figure 3.5 Maximum-Energy Lunar Trajectory for $t_f = 16$ hours

hours and final mass in LLO of 86% of the initial LEO mass. The optimal control for this Lunar spiral is shown in Figure 3.6.

The time-of-flight and velocity components can also be plotted versus initial Lunar radial distance for the Lunar maximum-energy problem and are shown in Figures 3.7, 3.8, and 3.9. Because of the need for prescribing the final mass in LLO, each figure shows several lines of constant final mass.

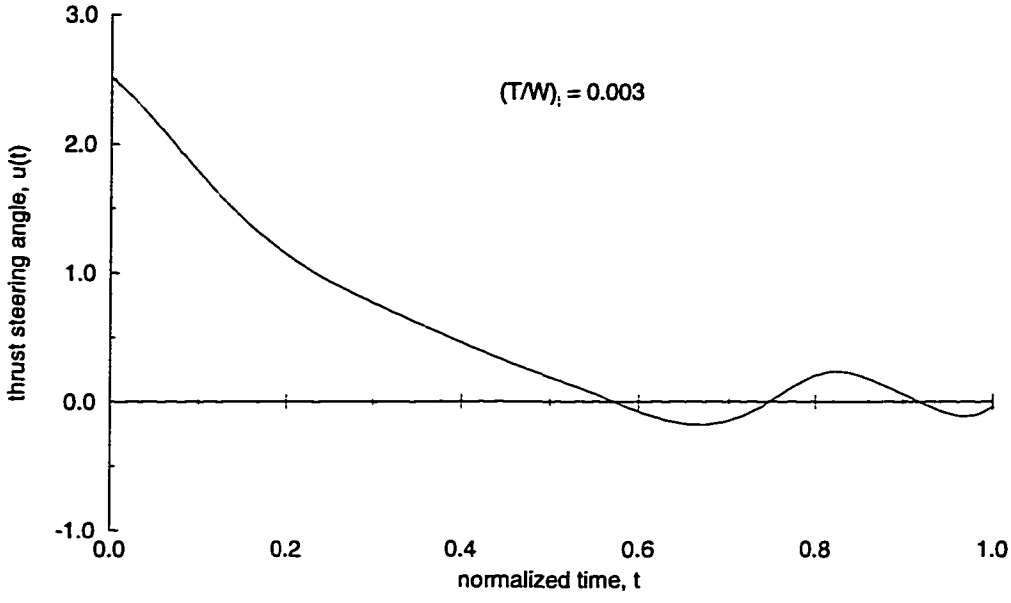

Figure 3.6 Maximum-Energy Optimal Control for Lunar Trajectory

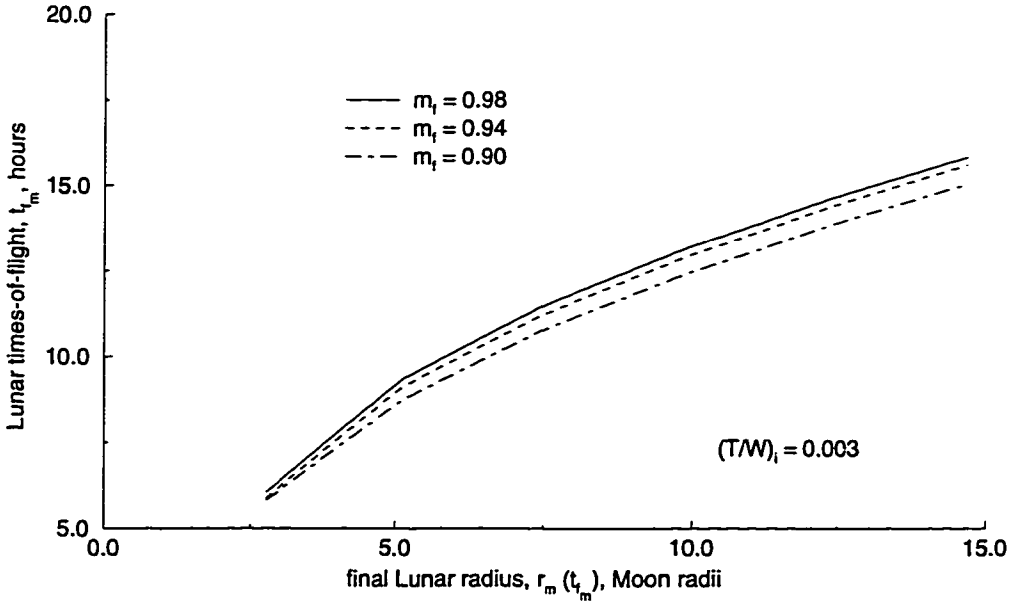

Figure 3.7 Flight Time of Lunar Spiral vs. Final Lunar Radial Distance

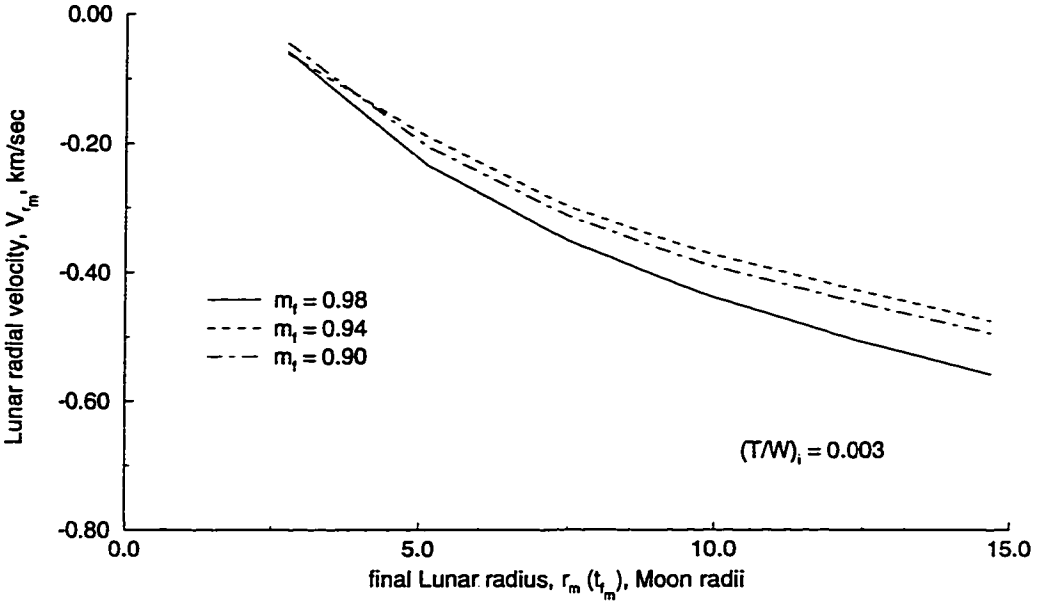

Figure 3.8 Final Radial Velocity of Lunar Spiral vs. Final Lunar Radial Distance

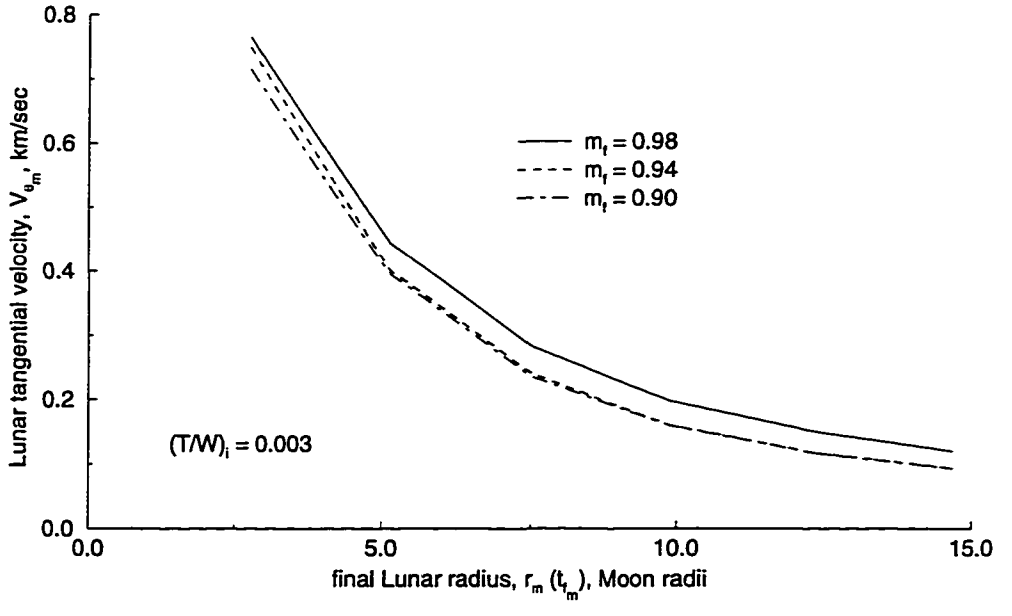

Figure 3.9 Final Tangential Velocity of Lunar Spiral vs. Final Lunar Radial Distance

Coast Patch Problem

The maximum-energy solutions of the previous section are used to provide boundary conditions for the translunar coast. The setup and solution of this translunar coast problem are presented in this section. The equations of motion are based on the circular restricted three-body problem [29]. This dynamic system assumes the Moon travels in a circular orbit about the Earth and that the mass of the spacecraft is negligible with respect to the Earth and Moon. The polar coordinate equations are derived in a rotating frame such that the Moon appears stationary with respect to the Earth in that frame. Figures given in this section and the following chapters which use these rotating coordinates will be plotted in that frame for clarity. The problem statement for the translunar coast can be stated as:

Find r_o (the radius at the beginning of the coast phase), θ_o (the polar angle at the beginning of the coast phase), m_f (the final mass), and the time-of-flight t_{coast} which minimize

$$J = t_{departure} + t_{capture} \tag{3.36}$$

subject to

$$\dot{r} = V_r \tag{3.37}$$

$$\dot{\theta} = \frac{V_\theta}{r} \tag{3.38}$$

$$\dot{V}_r = \frac{V_\theta^2}{r} - \frac{\mu_e}{r^2} + \mu_m \cos\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - \mu_m \frac{r}{r_m^3} + \omega^2 r \tag{3.39}$$

$$+2\omega V_\theta \tag{3.40}$$

$$\dot{V}_\theta = -\frac{V_r V_\theta}{r} - \mu_m \sin\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - 2\omega V_r \tag{3.41}$$

$$\tag{3.42}$$

where $r_m = \sqrt{r^2 + d^2 - 2rd\cos\theta}$, μ_e is the Earth gravity parameter, μ_m is the Moon gravity parameter, d is the distance between the Earth and Moon, ω is the radial velocity

of the Moon about the Earth, and the following relationships hold:

$$t_{departure} = t_{departure}(r_o) \tag{3.43}$$

$$t_{capture} = t_{capture}(r(t_{coast}), m_f) \tag{3.44}$$

The boundary conditions are

$$r(t_o) = r_o \tag{3.45}$$

$$\theta(t_o) = \theta_o \tag{3.46}$$

$$V_r(t_o) = f_1(r_o) \tag{3.47}$$

$$V_\theta(t_o) = f_2(r_o) \tag{3.48}$$

$$V_r(t_{coast}) = g_1(r(t_{coast}), m_f) \tag{3.49}$$

$$V_\theta(t_{coast}) = g_2(r(t_{coast}), m_f) \tag{3.50}$$

The objective function attempts to minimize the total time spent in the thrusting portions of the Lunar transfer, i.e., the Earth departure and Lunar capture. This has the effect of minimizing the total fuel used, since the assumption of a constant mass flow made for the maximum-energy problems yields a linear relationship between the mass and time. Thus, minimizing the time spent in a thrusting stage will also minimize the fuel used. The problem produces a sub-optimal solution since the Earth departure and Lunar capture are assumed to be maximum-energy spirals and there is no guarantee that these types of trajectories will produce an overall minimum-fuel solution. The trajectory is numerically integrated using the Bulirsch-Stoer integration scheme mentioned earlier. Equations (3.43) and (3.44) and Equations (3.47) - (3.50) are all functions of the radius at their respective points and possibly the final mass. These equations are determined from the maximum-energy spirals. Two methods are available for determining the way of forming these relationships. Pierson and Kluever [26] use a curve-fitting method where several max-energy spirals are solved and their solutions stored for both

the Earth departure and Lunar arrival. A variable order polynomial is then used for Equations (3.43), (3.47), (3.48) where the order of the polynomial is determined by the desired accuracy of the unknown point which is being determined. A two-dimensional cubic-spline is required for Equations (3.44), (3.49), and (3.50) since each requires two arguments, a radius and mass, to uniquely determine a new point. The method works quite well once the tabular data is determined. However, if any changes are made to the thrust, mass flow, initial and/or final conditions, a completely new table is needed. This disadvantage limits the usefulness of this method for a general methodology of solving minimum-fuel Lunar transfer problems. The next section describes a new dynamic boundary evaluation (DBE) [27] method which overcomes these problems.

Dynamic Boundary Evaluation

The basic idea of the DBE method is to satisfy the boundary conditions of the coast patch problem dynamically or on-line rather than use a static table. For example, when the coast patch problem wishes to know the maximum-energy spiral which terminates at a radius of 12 Earth radii, the DBE method will solve for that exact spiral and return the solution. The main advantage of the DBE method is that it eliminates the necessity of constructing a table of flight times and velocity components for the boundary conditions of the coast-patch problem. The construction of such a table is very labor intensive and can lead to the introduction of extraneous errors. Also, the time and effort required to construct such a table for specific trajectory flight parameters does not help in the solution of other problems with different parameters. Changing the initial thrust-to-weight ratio, for example, requires the formation of an entirely new table which cannot be formed from information contained in the old table. The same can be said of changes in the initial and final circular parking orbit altitudes and the initial mass. A second advantage of the DBE method is that it reduces the overall time required to solve the coast-patch problem. For the tabular method, the total time includes both the time

to construct the table and the processing time of finding the solution once the table is available. This can vary a great deal depending upon any prior knowledge of the solution. The table can be smaller and use a finer grid if the general range of the solution is known. However, if no knowledge of the solution is available, the table must be relatively large and use a coarse grid to insure that the solution's boundary conditions lie within the tabular values. The DBE method does not require any time other than that required for solving the maximum-energy problem. Also, the accuracy of the boundary conditions will be assured as long as it is possible to find a viable solution for the given input parameter.

I first tried to solve these boundary conditions on-line using a root-finding process [29]. The maximum-energy problem mentioned earlier is for a fixed final time and free final radius. The problem of finding the maximum-energy spiral which ends at a specified radius is equivalent to finding the time-of-flight, $t_{departure}$, which satisfies:

$$r_{desired} - r\left(t_{departure}\right) = 0 = F\left(t_{departure}\right) \tag{3.51}$$

The expression $r\left(t_{departure}\right)$ denotes the final radius found by solving a maximum-energy problem with a time-of-flight of $t_{departure}$. Equation (3.51) requires evaluating a function, however complicated, which is the difference between the desired final radius and the calculated final radius. The departure time which satisfies this equation completely determines that trajectory, and the result can be used as the initial conditions for the coast-patch problem, Equations (3.36) - (3.50). An iterative root-finding algorithm can then be used to solve Equation (3.51). This method proved effective but its performance was limited by the necessary computational burden of calculating a large number of maximum-energy spiral solutions for each iteration of the optimization. A new method of finding the boundary conditions by incorporating the root-finding process into the optimal solution of the approximate translunar problem was then derived.

A still more direct approach is to treat the radius, r, as the independent variable

rather than time. Then, the final radius is automatically the desired one and we have a fixed "end-time" problem. This conversion is possible since the radius is a non-decreasing function for the maximum-energy spiral. However, other trajectories near the maximum-energy solution may not have this same property, and the transformed differential equations can therefore become numerically unstable. These trajectories are usually computed iteratively which allows the possibility of a non-viable trajectory being considered. Also, the transformed equations of motion are singular for $r = r_o$ since the initial radial velocity is zero for circular target orbits. Numerical experiments starting the iterative process at a viable trajectory and with an extremely small non-zero $V_r(r_o)$ have not been encouraging, although an analytical asymptotic treatment remains a possible topic for future research.

An implementation of the previously mentioned iterative on-line method requires an optimization solution to be found within another optimization problem. These types of embedded optimal problems require special consideration and are difficult to solve using the traditional FORTRAN 77 programming language, because FORTRAN does not allow recursive function calls. It is necessary to either use another optimization program or to change the way the compiler creates the executable code. Since neither of these options is desirable, a different language, specifically C and then C++, was used for all of the numerical calculations. Care must also be taken that the embedded optimal problem, here the maximum-energy spiral, has a reasonable chance of convergence and that a "restart" is allowed. This is the primary difficulty involved in implementing the DBE method and the major cause of why the DBE method may not converge to a solution for the overall problem.

The solution of the maximum-energy problem usually involves solving the two-point boundary value problem obtained by applying optimal control theory which results in Equations (3.31) - (3.34) for the terminal boundary conditions on the costate. These

equations are repeated below for this discussion:

$$\lambda_r(t_{depart}) = -\frac{\mu}{r^2(t_{depart})} \tag{3.52}$$

$$\lambda_\theta(t_{depart}) = 0 \tag{3.53}$$

$$\lambda_{V_r}(t_{depart}) = -V_r(t_f) \tag{3.54}$$

$$\lambda_{V_\theta}(t_{depart}) = -V_\theta(t_f) \tag{3.55}$$

A similar boundary value problem can be formed by analyzing a fixed-final-radius maximum-speed optimal problem:

Given $r_{desired}$, find $u(\tau)$, $0 \leq \tau \leq 1$, and α which minimize

$$J = -\frac{V_r^2 + V_\theta^2}{2} \tag{3.56}$$

subject to

$$\frac{dr}{d\tau} = \alpha V_r \tag{3.57}$$

$$\frac{d\theta}{d\tau} = \alpha\frac{V_\theta}{r} \tag{3.58}$$

$$\frac{dV_r}{d\tau} = \alpha\left(\frac{V_\theta^2}{r} - \frac{\mu}{r^2} + a\sin u\right) \tag{3.59}$$

$$\frac{dV_\theta}{d\tau} = \alpha\left(-\frac{V_r V_\theta}{r} + a\cos u\right) \tag{3.60}$$

and

$$\Psi = r(1) - r_{desired} = 0 \tag{3.61}$$

where

$$a = \frac{T}{m_o - \beta\alpha\tau} \tag{3.62}$$

The control parameter α is the time-of-flight. The time transformation, $t = \alpha\tau$, allows us to convert a variable end-time problem in t into a fixed end-time problem in τ. The boundary conditions on the costate variables for this problem, obtained after applying optimal control theory, are:

$$\lambda_r(1) = \nu - \frac{\mu}{r_1^2} \tag{3.63}$$

$$\lambda_\theta(1) = 0 \tag{3.64}$$

$$\lambda_{V_r}(1) = -V_r(1) \tag{3.65}$$

$$\lambda_{V_\theta}(1) = -V_\theta(1) \tag{3.66}$$

where ν is a constant multiplier. The boundary conditions differ from Equations (3.52) - (3.55) only in the final value of the radial costate variable, λ_r. The resulting trajectory of either the maximum-energy spiral or restricted maximum-speed problem is completely determined by the costate boundary conditions and the time of flight parameter α. The maximum-speed problem can then be used to solve the maximum-energy problem with a specified final radius by enforcing Equations (3.61), (3.64), (3.65), and (3.66).

A similar type of maximum-speed problem can be formed for the lunar capture boundary conditions with the only added complexity being the unknown quantity of the final mass. Two basic ways of dealing with this have been implemented. The first is to treat the final mass as an additional design parameter and add a corresponding constraint to ensure that the linear mass equation is satisfied for the entire trajectory. This method is necessary when tables are used for the boundary condition evaluation since it is the only way to determine the final mass accurately. The second way of dealing with the final mass, which becomes possible when the boundary conditions are evaluated on-line, is to calculate the mass when needed using the linear mass relation. This can be done because the mass at the beginning of the lunar capture is known since the exact Earth departure is now known. This has the advantage of decreasing the number of design variables and constraints. However, the technical difficulties in implementing this outweigh any improvement in performance. Therefore, for simplicity, the first method of adding an extra design variable and constraint is used for this work.

The actual implementation of the DBE method uses both the on-line solution to the spirals and a form of the tabular method. Past solutions found by the DBE method are stored and, before a new solution is computed, the past solutions are checked to do two

things:

1. Check to see if the problem has already been solved and, if so, return that solution
2. Interpolate among the past solutions using a variable order interpolation scheme with a specified allowable error. If the interpolated value satisfies the error requirement, return that as the solution.

These checks eliminate repetitive solutions and greatly improves the speed of solving the coast patch problem. The implementation is then a "hybrid" of both the DBE and tabular methods. The DBE method and its implementation lends itself remarkably well to object-orientated programming and a class library. The use of such a library greatly simplifies the solution of the coast patch problem and other similar trajectories. A description of that library and how the DBE method is incorporated is given in Chapter 2.

Numerical Results

The coast patch problem is solved using the four control variables, initial radius, initial polar angle, time-of-flight, and final mass, as the design parameters for an NLP algorithm and enforcing the boundary conditions as constraints. The objective function is then the sum of the times-of-flight returned by the DBE method for the departure and capture. The solution of the translunar coast problem is very sensitive to the initial estimate of the design variables. A further complicating factor is the existence of both a posigrade and retrograde solution for a given coast transfer. Initial estimates are obtained from Rivas [29] for a range of initial thrust-to-weight ratios, and both the posigrade and retrograde solutions were found for each ratio. The essential difference between the posigrade and retrograde solutions is the sign of the tangential velocity with respect to the moon in LLO. A positive value indicates a posigrade solution and a negative value indicates a retrograde solution.The majority of solutions found in Rivas [29]

resulted in a posigrade solution. The corresponding retrograde solution was found by varying the initial radius and the initial polar angle at the beginning of the coast. The first step used was to increase the initial radius by several Earth radii and the initial polar angle anywhere from seventy to one hundred degrees. This resulted in the convergence to a retrograde solution for all of the lower thrust-to-weight ratios, i.e., those from $3(10^{-1})$ to $3(10^{-2})$ using NEP engines. The higher ratios, those from $5(10^{-2})$ to 0.5 using NTR engines, were much more difficult to solve for both solutions. The primary difficulty was the very small firing time required in the Lunar SOI to achieve the desired LLO. This firing time was never more than 14 minutes. This made the coast patch problem very sensitive to variations in the initial conditions, since the coast needed to place the spacecraft very close to the LLO for convergence to occur. This problem was alleviated by the knowledge, from Rivas [29], of a few retrograde solutions. These solutions were interpolated to estimate the design parameters that would converge to either a posigrade or retrograde capture. Every new solution would add to the known set and allow the interpolation to be carried out to higher-orders which increased the convergence rate for subsequent problems. Some additional variations were necessary for the first two or three solutions to be found, but once a sufficiently large number of known solutions were in each set, the solutions for the rest of the ratios were obtained easily.

Two initial thrust-to-weight ratio coast patches are detailed in Table 3.1 to describe the solutions. The trend between the posigrade and retrograde mass performance ratios

Table 3.1 Coast Patch Example Solutions

$(T/W)_i$	r_o, R_{earth}	θ_o, deg	time-of-flight, days	m_f/m_i	Solution
0.003	12.497	-35.18	4.58	0.93108	posigrade
0.003	12.693	-26.78	5.02	0.93024	retrograde
0.100	2.1687	326.10	5.10	0.63586	posigrade
0.100	2.2243	344.02	5.21	0.63552	retrograde

continues for all of the examined ratios, i.e., the posigrade solution always provides a better final mass than the corresponding retrograde solution.

Optimal Lunar Transfers

The full minimum-fuel Lunar transfer solution is found by a direct method which utilizes several features of optimal control from an indirect method. The general problem statement for the LEO-to-LLO transfer with a fixed thrust-coast-thrust engine sequence can be stated as:

Find the thrust direction time history $u(t)$, $t_o \leq t \leq t_f$, and t_{dep} (the time of the departure or the time of the first thrusting period), t_{coast} (the time of the translunar coast), and t_{cap} (the time of the capture or the time of the second thrusting period) which minimizes

$$J = -m(t_f) = (t_f - t_{cap}) + (t_{dep} - t_o) \tag{3.67}$$

subject to

$$\dot{r} = V_r \tag{3.68}$$

$$\dot{\theta} = \frac{V_\theta}{r} \tag{3.69}$$

$$\dot{V}_r = \frac{V_\theta^2}{r} - \frac{\mu_e}{r^2} + \mu_m \cos\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - \mu_m \frac{r}{r_m^3} + \omega^2 r \tag{3.70}$$

$$+2\omega V_\theta + a \sin u \tag{3.71}$$

$$\dot{V}_\theta = -\frac{V_r V_\theta}{r} - \mu_m \sin\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - 2\omega V_r + a \cos u \tag{3.72}$$

for $t_o \leq t \leq t_{dep}$ and where $r_m = \sqrt{r^2 + d^2 - 2rd\cos\theta}$ and subject to

$$\dot{r} = V_r \tag{3.73}$$

$$\dot{\theta} = \frac{V_\theta}{r} \tag{3.74}$$

$$\dot{V}_r = \frac{V_\theta^2}{r} - \frac{\mu_e}{r^2} + \mu_m \cos\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - \mu_m \frac{r}{r_m^3} + \omega^2 r \tag{3.75}$$

$$+2\omega V_\theta \tag{3.76}$$

$$\dot{V}_\theta = -\frac{V_r V_\theta}{r} - \mu_m \sin\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - 2\omega V_r \tag{3.77}$$

for $t_{dep} < t < t_{cap}$ and subject to

$$\dot{r}_m = V_{r_m} \tag{3.78}$$

$$\dot{\theta}_m = \frac{V_{\theta_m}}{r_m} \tag{3.79}$$

$$\dot{V}_{r_m} = \frac{V_{\theta_m}^2}{r_m} - \frac{\mu_m}{r_m^2} - \mu_e \frac{d\cos\theta_m + r_m}{r^3} + \mu_e \frac{\cos\theta_m}{d^2} + \omega^2 r_m \tag{3.80}$$

$$+2\omega V_{\theta_m} + a\sin u \tag{3.81}$$

$$\dot{V}_{\theta_m} = -\frac{V_{r_m} V_{\theta_m}}{r_m} + \mu_e \sin\theta_m \left(\frac{d}{r^3} - \frac{1}{d^2} \right) - 2\omega V_{r_m} + a\cos u \tag{3.82}$$

for $t_{cap} \leq t \leq t_f$ and where $r = \sqrt{r_m^2 + d^2 + 2 r_m d \cos\theta_m}$. The required boundary conditions are

$$r(t_o) = r_{LEO} \tag{3.83}$$

$$V_r(t_o) = 0 \tag{3.84}$$

$$V_\theta(t_o) = \sqrt{\mu_e / r_{LEO}} - \omega r_{LEO} \tag{3.85}$$

$$\bar{r}(t_{coast}) = P(\bar{r}_m(t_{coast})) \tag{3.86}$$

$$r_m(t_f) = r_{LLO} \tag{3.87}$$

$$V_{r_m}(t_f) = 0 \tag{3.88}$$

$$V_{\theta_m}(t_f) = \pm\sqrt{\mu_m / r_{LLO}} - \omega r_{LLO} \tag{3.89}$$

where $\bar{r} = [\, r,\ \theta,\ V_r,\ V_\theta \,]$ and $\bar{r}_m = [\, r_m,\ \theta_m,\ V_{r_m},\ V_{\theta_m} \,]$. The problem then is to find the thrust steering angle, $u(t)$, for the Earth departure and Lunar capture which uses the minimum amount of fuel to go from a low-Earth circular orbit, Equations (3.83) - (3.85), to a low-Lunar circular orbit, Equations (3.87) - (3.89). Equations (3.68) - (3.72) represent the three-body Earth centered equations with thrusting, Equations (3.73) - (3.77) represent the three-body Earth centered equations without thrusting, and Equations (3.78) - (3.82) represent the three-body Lunar-centered equations with thrusting.

Equation (3.89) will determine if a posigrade or retrograde solution is found depending upon the sign. Alternately, an absolute value of the velocity can be found which will allow the NLP algorithm to determine which type of capture is performed. The numerical results, which follow in the next section, show two sets of solutions one for each possible capture. Equation (3.86) matches the boundary conditions between the Earth-centered reference frame and the Lunar-centered frame, where $\bar{r}$ is a vector of the Earth-centered states and $\bar{r}_m$ is a vector of the Lunar-centered states, and P represents a conversion between the Earth and Lunar centered reference frames. The conversion from one frame to another is accomplished by the relations [29]:

$$r_m = \left(r^2 + d^2 - 2rd\cos\theta_e\right)^{1/2} \tag{3.90}$$

$$\theta_m = \arcsin\left(\frac{r_e \sin\theta_e}{r_m}\right) \tag{3.91}$$

$$if \quad (r_e\cos\theta_e - d < 0) \quad \theta_m = \pi - \theta_m \tag{3.92}$$

$$\begin{bmatrix} V_{r_m} \\ V_{\theta_m} \end{bmatrix} = \frac{1}{r_m}\begin{bmatrix} r_e - d\cos\theta_e & d\sin\theta_e \\ -d\sin\theta_e & r_e - d\cos\theta_e \end{bmatrix}\begin{bmatrix} V_r \\ V_\theta \end{bmatrix} \tag{3.93}$$

Therefore, the P operator applies the above relations to the given vector.

The thrust steering angle, $u(t)$, is found using optimal control theory. Since there are two powered portions of the trajectory, the Earth departure and the Lunar capture, a separate steering angle is obtained for each portion. Also, because the optimal steering angle requires a costate system, two costate systems are also derived. The Earth departure equations, Equations (3.68) - (3.72), will be denoted by $f = f(r, u)$ and the Lunar capture equations, Equations (3.78) - (3.82), will be denoted by $f_m = f_m(r_m, u)$.

The stationarity condition for the Earth departure, or first thrusting period, is given by:

$$\left(\frac{\partial f}{\partial u}\right)^T \lambda = 0 \tag{3.94}$$

which yields a tangent steering law,

$$\tan u^* = \frac{\lambda_{V_r}}{\lambda_{V_\theta}} \tag{3.95}$$

$$\Rightarrow \sin u = \frac{\pm\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.96}$$

$$\cos u = \frac{\pm\lambda_{V_\theta}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.97}$$

$$\tag{3.98}$$

The Legendre-Clebsch condition provides the sign:

$$0 \leq H_{uu} \tag{3.99}$$

$$\leq -a\left(\lambda_{V_r}\sin u + \lambda_{V_\theta}\cos u\right) \tag{3.100}$$

$$\leq -a\left(\frac{\lambda_{V_r}\left(\pm\lambda_{V_r}\right) + \lambda_{V_\theta}\left(\pm\lambda_{V_\theta}\right)}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}}\right) \tag{3.101}$$

Therefore the negative sign is needed, i.e.:

$$\sin u^* = \frac{-\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.102}$$

$$\cos u^* = \frac{-\lambda_{V_\theta}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{3.103}$$

The costate system for this portion of the transfer is given by:

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial r}\right)^T \lambda \tag{3.104}$$

The Jacobian is found to be

$$\frac{\partial f}{\partial r} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -\frac{V_\theta}{r^2} & 0 & 0 & \frac{1}{r} \\ \frac{\partial \dot{V}_r}{\partial r} & \frac{\partial \dot{V}_r}{\partial \theta} & 0 & \frac{2V_\theta}{r} + 2\omega \\ \frac{\partial \dot{V}_\theta}{\partial r} & \frac{\partial \dot{V}_\theta}{\partial \theta} & -\frac{V_\theta}{r} - 2\omega & -\frac{V_r}{r} \end{bmatrix} \tag{3.105}$$

where the partials are:

$$\frac{\partial \dot{V}_r}{\partial r} = -\frac{V_\theta^2}{r^2} + \frac{2\mu_e}{r^3} + \frac{\mu_m}{r_m^5}\left[3\left(r - d\cos\theta\right)^2 - r_m^2\right] + \omega^2 \tag{3.106}$$

$$\frac{\partial \dot{V}_r}{\partial \theta} = \mu_m \frac{d\sin\theta}{r_m^5}\left[3r\left(r - d\cos\theta\right) - r_m^2\right] + \mu_m \frac{\sin\theta}{d^2} \tag{3.107}$$

$$\frac{\partial \dot{V}_\theta}{\partial r} = \frac{V_r V_\theta}{r^2} + 3\mu_m d \sin\theta \frac{r - d\cos\theta}{r_m^5} \tag{3.108}$$

$$\frac{\partial \dot{V}_\theta}{\partial \theta} = -\mu_m \cos\theta \left(\frac{d}{r_m^3} - \frac{1}{d^2}\right) + 3\mu_m r \frac{d^2 \sin^2\theta}{r_m^5} \tag{3.109}$$

The costate equations for the Earth departure are found by using the Jacobian and Equations (3.102) and (3.103) in Equation (3.104):

$$\dot{\lambda}_r = \frac{V_\theta}{r^2}\lambda_\theta + \left\{\frac{V_\theta^2}{r^2} - \frac{2\mu}{r^3} - \frac{\mu_m}{r_m^5}\left[3\left(r - d\cos\theta\right)^2 - r_m^2\right] - \omega^2\right\}\lambda_{V_r} \tag{3.110}$$
$$- \left\{\frac{V_r V_\theta}{r^2} + 3\mu_m d \sin\theta \frac{r - d\cos\theta}{r_m^5}\right\}\lambda_{V_\theta}$$

$$\dot{\lambda}_\theta = -\left\{\mu_m \frac{d\sin\theta}{r_m^5}\left[3r\left(r - d\cos\theta\right) - r_m^2\right] + \mu_m \frac{\sin\theta}{d^2}\right\}\lambda_{V_r} \tag{3.111}$$
$$- \left\{-\mu_m \cos\theta\left(\frac{d}{r_m^3} - \frac{1}{d^2}\right) + 3\mu_m r \frac{d^2\sin^2\theta}{r_m^5}\right\}\lambda_{V_\theta}$$

$$\dot{\lambda}_{V_r} = -\lambda_r + \left(\frac{V_\theta}{r} + 2\omega\right)\lambda_{V_\theta} \tag{3.112}$$

$$\dot{\lambda}_{V_\theta} = -\frac{\lambda_\theta}{r} - \left(\frac{2V_\theta}{r} + 2\omega\right)\lambda_{V_r} + \frac{V_r}{r}\lambda_{V_\theta} \tag{3.113}$$

The stationarity condition for the Lunar capture, or second thrusting period, is given by

$$\left(\frac{\partial f_m}{\partial u}\right)^T \lambda_m = 0 \tag{3.114}$$

which yields the tangent steering law,

$$\tan u^* = \frac{\lambda_{V_{r_m}}}{\lambda_{V_{\theta_M}}} \tag{3.115}$$

$$\Rightarrow \sin u = \frac{\pm\lambda_{V_{r_m}}}{\sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}} \tag{3.116}$$

$$\cos u = \frac{\pm\lambda_{V_{\theta_m}}}{\sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}} \tag{3.117}$$

$$\tag{3.118}$$

The Legendre-Clebsch condition provides the sign:

$$0 \leq H_{uu} \tag{3.119}$$

$$\leq -a\left(\lambda_{V_{r_m}}\sin u + \lambda_{V_{\theta_m}}\cos u\right) \tag{3.120}$$

$$\leq -a\left(\frac{\lambda_{V_{r_m}}\left(\pm\lambda_{V_{r_m}}\right) + \lambda_{V_{\theta_m}}\left(\pm\lambda_{V_{\theta_m}}\right)}{\sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}}\right) \tag{3.121}$$

Therefore the negative sign is needed, i.e.:

$$\sin u^* = \frac{-\lambda_{V_{r_m}}}{\sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}} \tag{3.122}$$

$$\cos u^* = \frac{-\lambda_{V_{\theta_m}}}{\sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}} \tag{3.123}$$

$$\tag{3.124}$$

The costate system for this portion of the transfer is given by:

$$\dot{\lambda}_m = -\left(\frac{\partial f_m}{\partial r_m}\right)^T \lambda_m \tag{3.125}$$

The Jacobian is found to be

$$\frac{\partial f_m}{\partial r_m} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -\frac{V_{\theta_m}}{r_m^2} & 0 & 0 & \frac{1}{r_m} \\ \frac{\partial \dot{V}_{r_m}}{\partial r_m} & \frac{\partial \dot{V}_{r_m}}{\partial \theta_m} & 0 & \frac{2V_{\theta_m}}{r_m} + 2\omega \\ \frac{\partial \dot{V}_{\theta_m}}{\partial r_m} & \frac{\partial \dot{V}_{\theta_m}}{\partial \theta_m} & -\frac{V_{\theta_m}}{r_m} - 2\omega & -\frac{V_{r_m}}{r_m} \end{bmatrix} \tag{3.126}$$

where the partials are:

$$\frac{\partial \dot{V}_{r_m}}{\partial r_m} = -\frac{V_{\theta_m}^2}{r_m^2} + \frac{2\mu_m}{r_m^3} - \frac{\mu_e}{r^5}\left[r^2 - 3\left(r_m + d\cos\theta_m\right)^2\right] + \omega^2 \tag{3.127}$$

$$\frac{\partial \dot{V}_{r_m}}{\partial \theta_m} = -\mu_e\frac{d\sin\theta_m}{r^5}\left[3r_m\left(r_m + d\cos\theta_m\right) - r^2\right] - \mu_e\frac{\sin\theta_m}{d^2} \tag{3.128}$$

$$\frac{\partial \dot{V}_{\theta_m}}{\partial r_m} = \frac{V_{r_m}V_{\theta_m}}{r_m^2} - 3\mu_e d\sin\theta_m\frac{r_m + d\cos\theta_m}{r^5} \tag{3.129}$$

$$\frac{\partial \dot{V}_{\theta_m}}{\partial \theta_m} = \mu_e\cos\theta_m\left(\frac{d}{r^3} - \frac{1}{d^2}\right) + 3\mu_e r_m\frac{d^2\sin^2\theta_m}{r^5} \tag{3.130}$$

The costate equations for the Lunar capture is found by using this Jacobian and Equations (3.122) and (3.123) in Equation (3.125):

$$\dot{\lambda}_{r_m} = \frac{V_{\theta_m}}{r_m^2}\lambda_{\theta_m} + \left\{\frac{V_{\theta_m}^2}{r_m^2} - \frac{2\mu_m}{r_m^3} + \frac{\mu_e}{r^5}\left[r^2 - 3\left(r_m + d\cos\theta_m\right)^2\right] - \omega^2\right\}\lambda_{V_{r_m}} \tag{3.131}$$

$$-\left\{\frac{V_{r_m}V_{\theta_m}}{r_m^2}-3\mu_e d\sin\theta_m\frac{r_m+d\cos\theta_m}{r^5}\right\}\lambda_{V_{\theta_m}}$$

$$\dot{\lambda}_{\theta_m} = \left\{\mu_e\frac{d\sin\theta_m}{r^5}\left[3r_m\left(r_m+d\cos\theta_m\right)-r^2\right]\mu_e\frac{\sin\theta_m}{d^2}\right\}\lambda_{V_{r_m}} \tag{3.132}$$

$$-\left\{\mu_e\cos\theta_m\left(\frac{d}{r^3}-\frac{1}{d^2}\right)+3\mu_e r_m\frac{d^2\sin^2\theta_m}{r^5}\right\}\lambda_{V_{\theta_m}}$$

$$\dot{\lambda}_{V_{r_m}} = -\lambda_{r_m}+\left(\frac{V_{\theta_m}}{r_m}+2\omega\right)\lambda_{V_{\theta_m}} \tag{3.133}$$

$$\dot{\lambda}_{V_{\theta_m}} = -\frac{\lambda_{\theta_m}}{r_m}-2\left(\frac{V_{\theta_m}}{r_m}+\omega\right)\lambda_{V_{r_m}}+\frac{V_{r_m}}{r_m}\lambda_{V_{\theta_m}} \tag{3.134}$$

Transversality conditions on the Lunar capture equations yields one useful costate boundary value. The final value of the Lunar costate vector is given by:

$$\lambda_m\left(t_f\right)=\phi+\left(\frac{\partial\psi}{\partial r_m}\right)^T\nu \tag{3.135}$$

where $\phi=-m\left(t_f\right)$ and

$$\psi=\begin{bmatrix} r_m\left(t_f\right)-r_{LLO} \\ V_{r_m}\left(t_f\right) \\ V_{\theta_m}\left(t_f\right)-\sqrt{\mu_m/r_m}+\omega r_{LLO}\end{bmatrix} \tag{3.136}$$

The terminal constraints above are identical to Equations (3.87) - (3.89). Inserting into Equation (3.135) and simplifying

$$\lambda_{r_m}\left(t_f\right) = \nu_1 \tag{3.137}$$

$$\lambda_{\theta_m}\left(t_f\right) = 0 \tag{3.138}$$

$$\lambda_{V_{r_m}}\left(t_f\right) = \nu_2 \tag{3.139}$$

$$\lambda_{V_{\theta_m}}\left(t_f\right) = \nu_3 \tag{3.140}$$

Equation (3.138) provides the useful boundary condition.

The method of solving the optimal transfer problem can now be summarized. Initial estimates of the initial polar angle and the unknown initial costates, $\lambda_r\left(t_o\right)$, $\lambda_{V_r}\left(t_o\right)$, and $\lambda_{V_\theta}\left(t_o\right)$ are used to integrate Equations (3.68)-(3.72) forward in time to $t=t_{dep}$. The

initial costate on the polar angle is zero, $\lambda_\theta = 0$, because the polar angle itself is free. Then Equations (3.73)-(3.77) are integrated from this point to $t = t_{coast}$. Estimates for the unknown final conditions for the Lunar polar angle, $\theta_m(t_f)$, final mass, $m(t_f)$, and unknown costates, $\lambda_{r_m}(t_f)$, $\lambda_{V_{r_m}}(t_f)$, and $\lambda_{V_{\theta_m}}(t_f)$ are then used to integrate Equations (3.78) - (3.82) backward in time from $t = t_f$ to $t = t_{cap}$. The boundary conditions of Equation (3.86) are then applied to match the terminal coast conditions to the initial capture conditions. The NLP algorithm then has eleven design variables: the initial and final polar angles, the initial costate values of the radius, radial velocity, and tangential velocity, the final costate values of the radius, radial velocity, and tangential velocity, and the times-of-flight for the Earth departure, translunar coast, and Lunar capture. Four equality constraints are implemented which are exactly Equation (3.86).

Another, perhaps more straight forward, method of solution would have the constraints directly enforced at the final time, i.e., prescribing a circular orbit about the Moon. This method again uses initial estimates of the initial polar angle and the unknown initial costates, $\lambda_r(t_o)$, $\lambda_{V_r}(t_o)$, and $\lambda_{V_\theta}(t_o)$, to integrate Equations (3.68) - (3.72) forward in time to $t = t_{dep}$. Then Equations (3.73) - (3.77) are integrated from this point to $t = t_{coast}$. The transformation between the Earth-centered and Lunar-centered equations is then performed, i.e., the transformation operator P. The result is used to integrate Equations (3.78) - (3.82) forward in time to $t = t_f$. The terminal constraints are applied (Equations (3.87) - (3.89)) and the costate boundary condition is applied, Equations (3.138). The NLP algorithm for this method has only seven design variables: the initial polar angle and costate values for the radius, radial velocity, and tangential velocity, and the times-of-flight for the departure, coast, and capture. Four equality constraints are implemented which enforce the three desired terminal conditions on the states requiring a circular orbit and the costate boundary conditions This method uses four less design variables but is much less robust than the previous method. Also, the transformation of the Earth-centered costate variables becomes extremely com-

plex. Because of these difficulties, the larger design vector method described initially is recommended.

Estimates for the times-of-flight and costate values are provided by the initial approximations discussed earlier in this chapter. This was done by finding the translunar coast solution for the desired initial thrust-to-weight ratio, LEO, LLO, and other mission parameters. This solution prescribes a specific maximum-energy spiral for the Earth departure and Lunar capture. These solutions yield the respective flight times and costate values. The initial and final polar angle are not known and must be estimated from the translunar coast. Initially, the number of spirals of the departure and/or capture was found and this was simply subtracted from desired polar angle position returned by the translunar coast solution. However, this proved to be unstable since the three-body effects on the maximum-energy spirals tended to "shrink" the spiral, i.e., the final polar angle from a three-body spiral always lagged its respective two-body solution. The solution to the translunar coast is sensitive to this polar angle perturbation and will not converge. The easiest method that I found for determining the initial polar angle which results in the desired final polar angle, returned from the translunar coast solution, and using the same flight time was to solve an auxiliary NLP problem. The solution finds a feasible point where the constraints are simply the desired final polar angle and other states from the maximum-energy solution. This auxiliary problem is extremely stable, can be easily solved, and can be stated as:

Find $\theta(t_o)$, $\lambda_r(t_o)$, $\lambda_{V_r}(t_o)$, and $\lambda_{V_\theta}(t_o)$, which satisfies:

$$\bar{r}(t_{dep}) - \bar{r}_{desired} = 0 \tag{3.141}$$

where $\bar{r}(t_{dep})$ is found by integrating Equations (3.68) - (3.72) from $t = t_o$ to $t = t_{dep}$ and $\bar{r}_{desired}$ is the vector of states required by the translunar coast solution. This auxiliary problem is really a multidimensional root-finding problem and can be solved by the feasibility option which was discussed in Chapter 2. The type of solution, posigrade or

retrograde, was determined by the initial estimate from the coast patch problem.

Numerical Results

A range of initial thrust-to-weight ratios was analyzed using the two different types of engines to accomplish the range. The low-end of the ratios ran from $1(10^{-3})$ to $3(10^{-2})$ in increments of 0.002 using electric propulsion. The ratios of the final mass over the initial mass, or the optimal mass performance ratio, with respect to the initial thrust-to-weight for the electric propulsion systems are given in Figure 3.10 for both

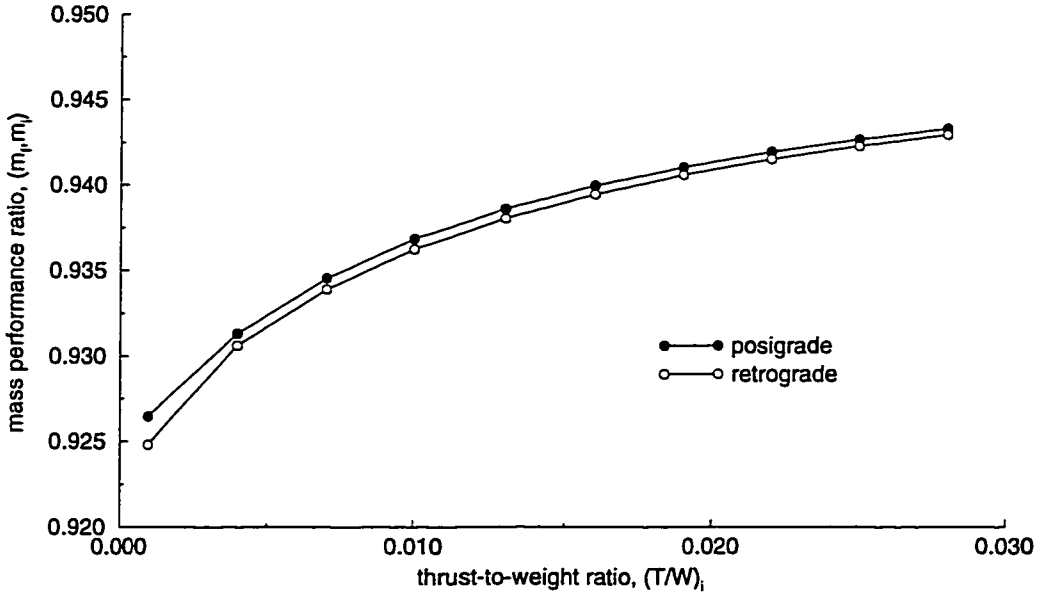

Figure 3.10 Final Mass Ratio vs Initial Thrust-to-Weight Ratio for NEP

types of Lunar captures. The lowest performance values occur at $(T/W)_i = 1(10^{-3})$ with a posigrade mass ratio of 0.933 and a retrograde mass ratio of 0.929. The highest values occur at $(T/W)_i = 3(10^{-2})$ with a posigrade mass ratio of 0.943 and a retrograde mass ratio of 0.9428.

The curves may at first appear to be counter intuitive since the performance ratio increases with increasing initial thrust-to-weight. However, the the total engine-on-times

of the trajectories decrease more rapidly than the increase of mass flow rate caused by the increase in thrust. Although the performance ratio increases indicating lower fuel requirements, the mass payload delivered to LLO does not increase with a larger thrust-to-weight ratio and a simple analysis shows why. The initial mass of the spacecraft in LEO is a combination of payload, propulsion system and its support, propellant, and other miscellaneous systems and structural support mass. The variation of the thrust-to-weight ratio is done by increasing the propulsive thrust rather than decreasing the mass, since a comparison between the transfers ultimately depends upon the amount of payload the transfer delivers rather than how "quickly" any propulsion system delivers a fixed desired payload. The propulsion system mass can be assumed to be proportional to the thrust [1] and the miscellaneous structural mass is proportional to the amount of fuel, i.e., more fuel requires larger tanks, pumps, etc. The change in payload mass delivered can then be viewed as a trade-off between a higher propulsion system weight versus a lower amount of fuel and accompanying structural weight. A comparison between the proportionality constants clearly shows that the propulsive weight increases much more quickly and that the payload delivered to LLO therefore decreases with increasing initial thrust-to-weight.

An example NEP posigrade trajectory for an initial thrust-to-weight ratio of $3\,(10^{-3})$, is shown in Figure 3.11 The posigrade transfer has a total trip time of 7.27 days, where 2.23 days are spent in the departure, 4.59 days in the translunar coast, and 10.65 hours are spent in the capture, and the corresponding final mass is 93,065 kg. The NEP retrograde trajectory for the same initial thrust-to-weight ratio is given in Figure 3.12. The retrograde transfer has a total trip time of 7.73 days where 2.24 days are spent in the departure, 4.71 days in the translunar coast, and 11.9 hours are spent in the capture. The final mass is 93,025 kg.

The upper $(T/W)_i$ ratios for NTR propulsion are 0.05 to 0.5 in increments of 0.02, and the respective mass performance ratios are given in Figure 3.13. The performance

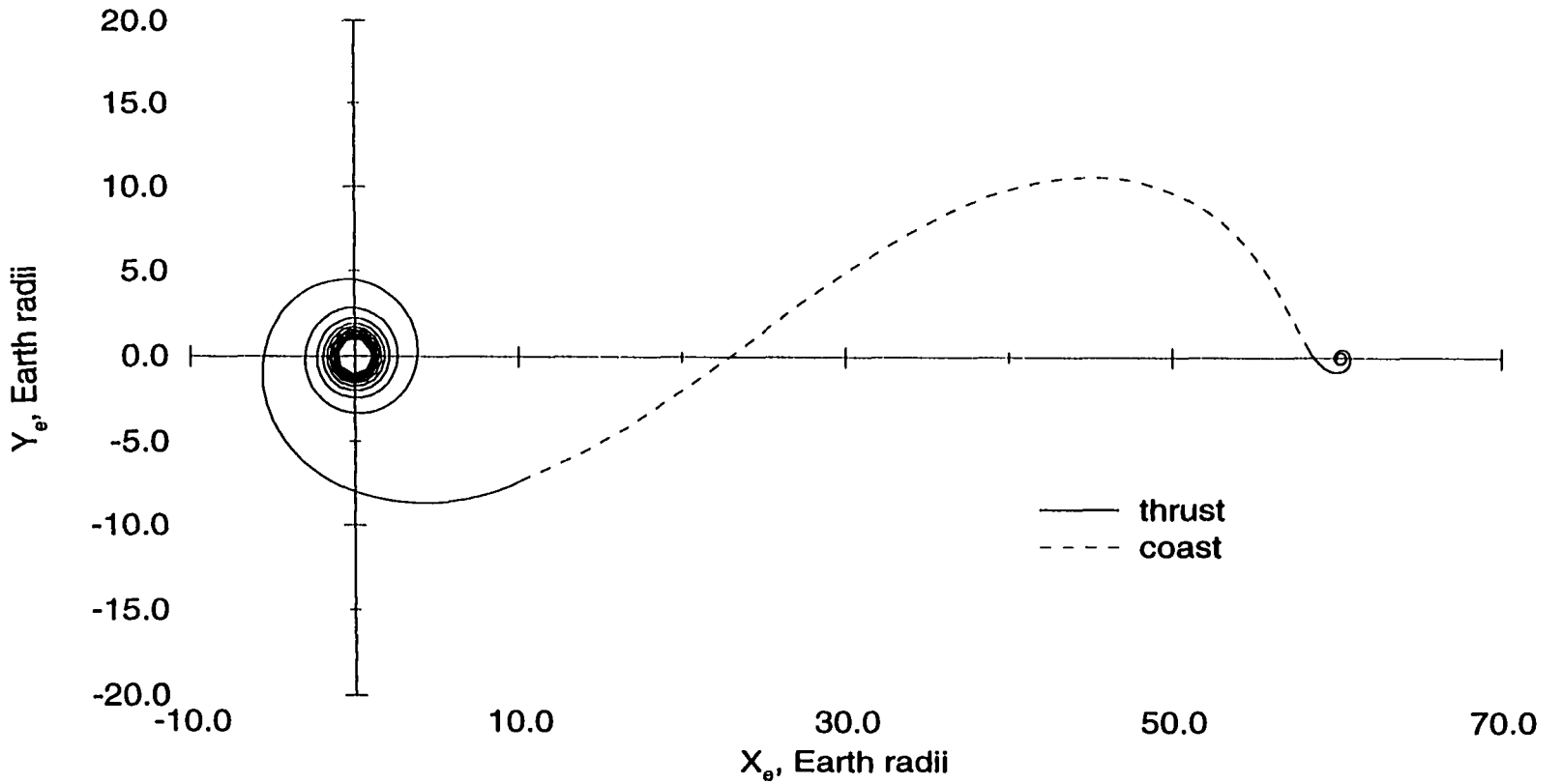

Figure 3.11 Posigrade Optimal Lunar Transfer for $(T/W)_i = 3(10^{-3})$

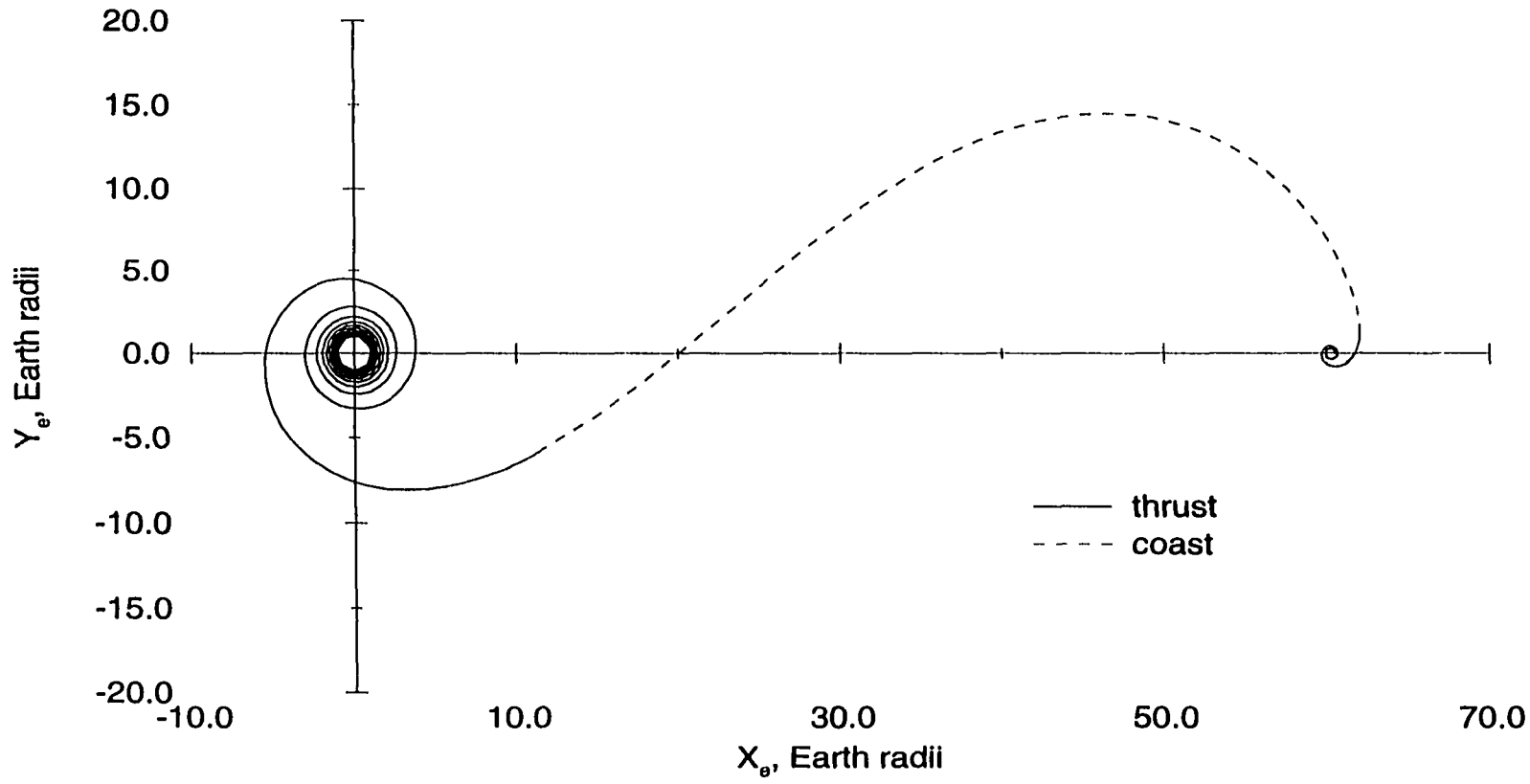

Figure 3.12 Retrograde Optimal Lunar Transfer for $(T/W)_i = 3\,(10^{-3})$

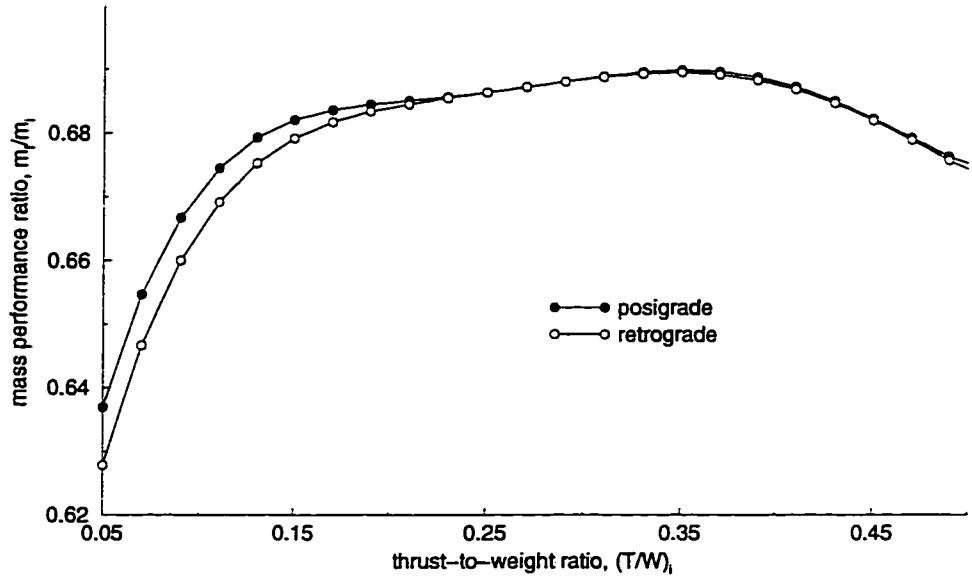

Figure 3.13 Final Mass Ratio vs Initial Thrust-to-Weight for NTR

curves differ greatly from the NEP solutions presented earlier. The curves for both types of capture achieve a maximum at $(T/W)_i = 0.35$ with a posigrade value of 0.6898 and a retrograde value of 0.6894. The posigrade and retrograde values of the mass ratios at the lowest thrust-to-weight are 0.6369 and 0.6278, respectively. The mass ratio difference between the captures is greatest at this point and smallest at the maximum point. A logical argument similar to the one discussed earlier for the NEP transfers, shows that the payload decreases for the NTR transfers in the same manner as for the NEP transfers including where the curves pass through their maximal point. The rated thrust for a NTR can be increased by adding additional fuel cells which add a corresponding weight. This additional weight is much greater than the propellant savings. The maximum is then a characteristic of the problem and not a result of some "ideal" payload weight. Similar results were found in Rivas [29] but those results were somewhat inconclusive. The previous work [29] did not use optimal control theory to steer the vehicle and, because of convergence problems and possible numerical noise, did not claim that this

maximum existed. However, these results confirm the previous results although the actual mass values are slightly different.

The posigrade NTR optimal transfers for an initial thrust-to-weight ratio of 0.1 is shown in Figure 3.14 This optimal transfer has a total trip time of 5.075 days where 55.145 minutes are spent in the departure, 4.983 days in the translunar coast, and 9.55 minutes are spent in the capture. The final mass is 67,107 kg. The retrograde NTR optimal transfer for the same initial thrust-to-weight ratio is given in Figure 3.15. This transfer has a total trip time of 4.943 days where 51.82 minutes are spent in the departure, 4.9 days in the translunar coast, and 9.589 minutes are spent in the capture. The final mass is 66,505 kg. Note the short time spent thrusting, especially for the Lunar captures.

Solution of the 2PBVP

The complete optimal solution of the Lunar transfer is found by directly solving the resulting 2PBVP. The solutions of the previous section are described as optimal, but do not directly solve the 2PBVP. This section shows that the previous solutions are optimal because they do solve the 2PBVP. The complete problem statement is derived and the method of its solution is presented. An overview of convergence properties is also presented. The solution of the 2PBVP would provide the most accurate solution possible for an optimal Lunar transfer.

The complete problem statement for a fixed thrust-coast-thrust firing sequence two-dimensional low-thrust Earth-Moon transfer is given by: Find $u_e(\tau)$, $u_m(\tau)$, α_e, α_c, and α_m, $0 \leq \tau \leq 1$ which minimize

$$J = \alpha_e + \alpha_m = \phi[x(t_f), \alpha] \tag{3.142}$$

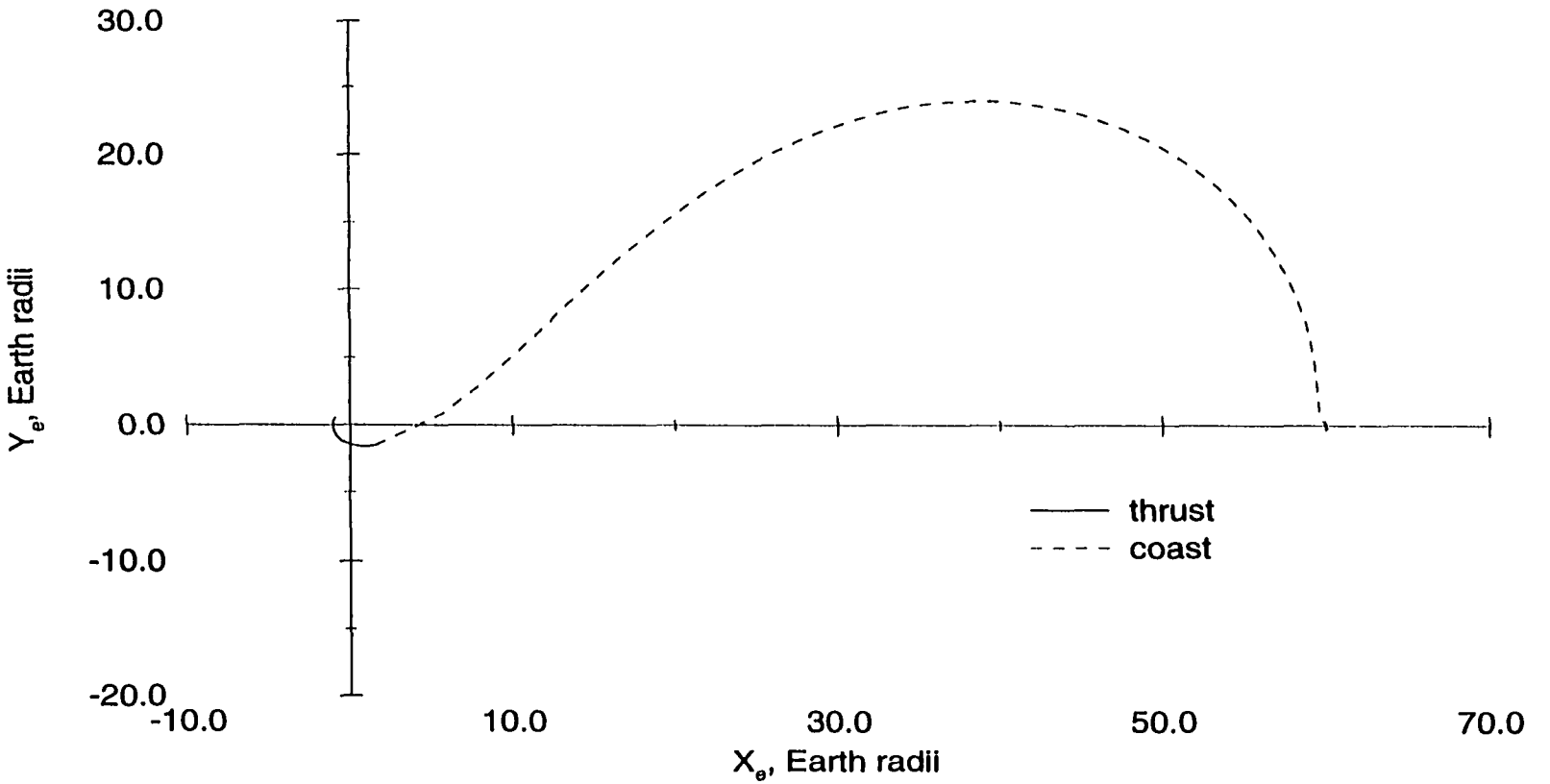

Figure 3.14 Posigrade Optimal Lunar Transfer for $(T/W)_i = 0.1$

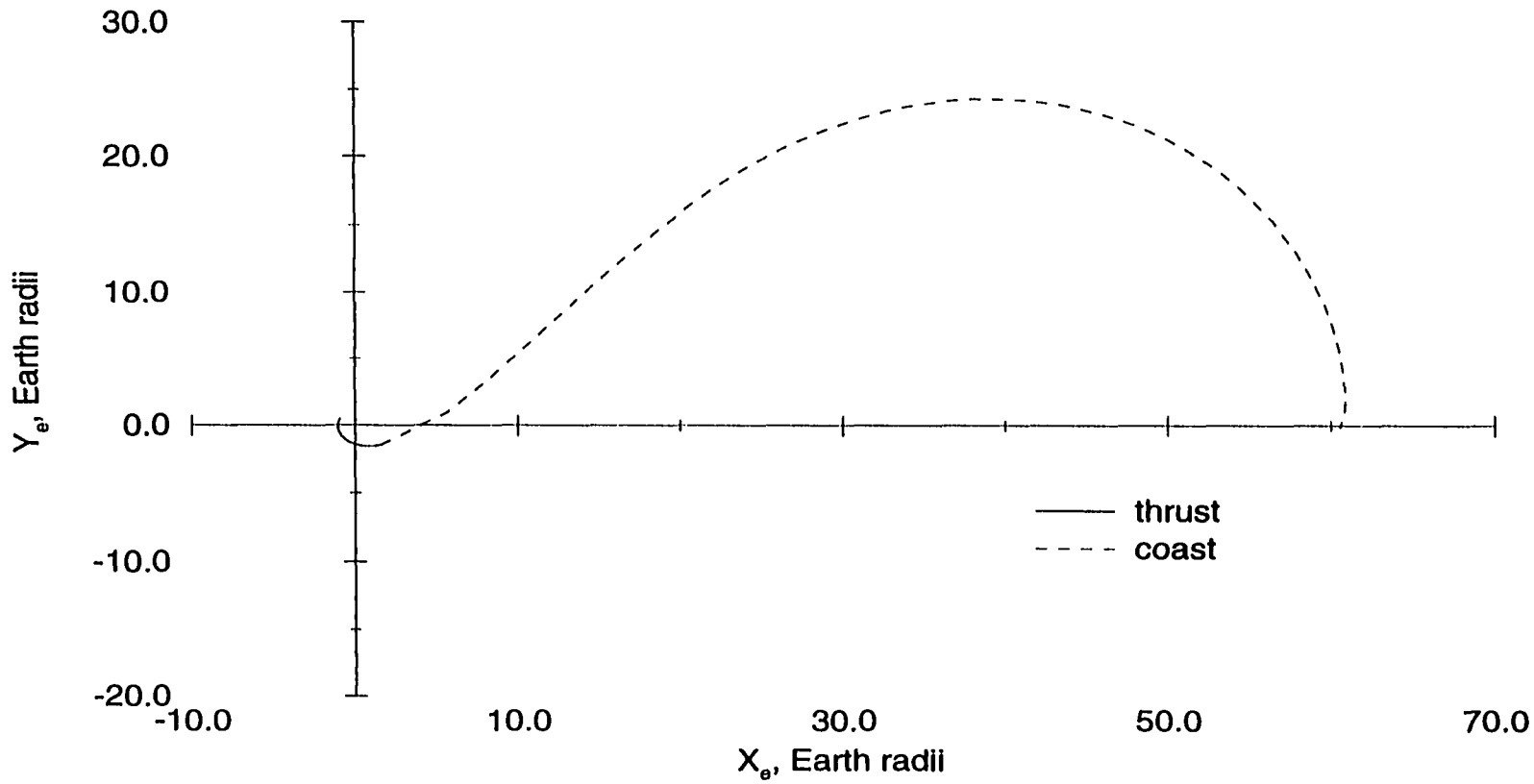

Figure 3.15 Retrograde Optimal Lunar Transfer for $(T/W)_i = 0.1$

subject to:

$$\frac{d\bar{r}_e}{d\tau} = \alpha_e f_e(\bar{r}_e, u_e) = F_e(\bar{r}_e, u_e) \tag{3.143}$$

$$\frac{d\bar{r}_c}{d\tau} = \alpha_c f_c(\bar{r}_c) = F_c(\bar{r}_c) \tag{3.144}$$

$$\frac{d\bar{r}_m}{d\tau} = \alpha_m f_m(\bar{r}_m, u_m) = F_m(\bar{r}_m, u_m) \tag{3.145}$$

where

$$\bar{r}_e = [r_e \;\; \theta_e \;\; V_{r_e} \;\; V_{\theta_e}]^T \tag{3.146}$$

$$\bar{r}_c = [r_c \;\; \theta_c \;\; V_{r_c} \;\; V_{\theta_c}]^T \tag{3.147}$$

$$\bar{r}_m = [r_m \;\; \theta_m \;\; V_{r_m} \;\; V_{\theta_m}]^T \tag{3.148}$$

and

$$f_e(\bar{r}_e, u_e) = \begin{bmatrix} V_{r_e} \\ \frac{V_{\theta_e}}{r_e} \\ \frac{V_{\theta_e}^2}{r_e} - \frac{\mu_e}{r_e^2} + \mu_m \frac{d\cos\theta_e - r_e}{r_m^3} - \mu_m \frac{\cos\theta_e}{d^2} + \omega^2 r_e + 2\omega V_{\theta_e} \\ \qquad +a\sin u_e \\ -\frac{V_{r_e} V_{\theta_e}}{r_e} - \mu_m \sin\theta_e \left(\frac{d}{r_m^3} - \frac{1}{d^2}\right) - 2\omega V_{r_e} + a\cos u_e \end{bmatrix} \tag{3.149}$$

$$f_c(\bar{r}_c) = \begin{bmatrix} V_{r_c} \\ \frac{V_{\theta_c}}{r_c} \\ \frac{V_{\theta_c}^2}{r_c} - \frac{\mu_e}{r_c^2} + \mu_m \frac{d\cos\theta_c - r_c}{r_m^3} - \mu_m \frac{\cos\theta_c}{d^2} + \omega^2 r_c + 2\omega V_{\theta_c} \\ -\frac{V_{r_c} V_{\theta_c}}{r_c} - \mu_m \sin\theta_c \left(\frac{d}{r_m^3} - \frac{1}{d^2}\right) - 2\omega V_{r_c} \end{bmatrix} \tag{3.150}$$

$$f_m(\bar{r}_m, u_m) = \begin{bmatrix} V_{r_m} \\ \frac{V_{\theta_m}}{r_m} \\ \frac{V_{\theta_m}^2}{r_m} - \frac{\mu_m}{r_m^2} - \mu_e \frac{d\cos\theta_m + r_m}{r_e^3} + \mu_e \frac{\cos\theta_m}{d^2} + \omega^2 r_m + 2\omega V_{\theta_m} \\ \qquad +a\sin u_m \\ -\frac{V_{r_m} V_{\theta_m}}{r_m} + \mu_e \sin\theta_m \left(\frac{d}{r_e^3} - \frac{1}{d^2}\right) - 2\omega V_{r_m} + a\cos u_m \end{bmatrix} \tag{3.151}$$

and satisfying:

$$\Psi\left[\bar{r}\left(0\right),\bar{r}\left(1\right)\right]=\begin{bmatrix}\psi_o\\ \psi_{e/c}\\ \psi_{c/m}\\ \psi_f\end{bmatrix}=0 \tag{3.152}$$

where

$$\psi_o = \begin{bmatrix} r_e\left(0\right)-r_o \\ V_{r_e}\left(0\right) \\ V_{\theta_e}\left(0\right)-\sqrt{\mu_e/r_o}+\omega r_o \end{bmatrix} \tag{3.153}$$

$$\psi_{e/c} = \bar{r}_c\left(0\right)-\bar{r}_e\left(1\right) \tag{3.154}$$

$$\psi_{c/m} = \begin{bmatrix} r_m\left(0\right)-\tilde{r} \\ \frac{\sin\theta_m(0)}{r_c(1)}-\frac{\sin\theta_c(1)}{\tilde{r}} \\ V_{r_m}\left(0\right)-\frac{1}{\tilde{r}}\left[\left(d-r_c\left(1\right)\cos\theta_c\left(1\right)\right)V_{r_c}\left(1\right)-d\sin\theta_c\left(1\right)V_{\theta_c}\left(1\right)\right] \\ V_{\theta_m}\left(0\right)-\frac{1}{\tilde{r}}\left[d\sin\theta_c\left(1\right)V_{r_c}\left(1\right)+\left(d-r_c\left(1\right)\cos\theta_c\left(1\right)\right)V_{\theta_c}\left(1\right)\right] \end{bmatrix} \tag{3.155}$$

$$\psi_f = \begin{bmatrix} r_m\left(1\right)-r_f \\ V_{r_m}\left(1\right) \\ V_{\theta_m}\left(1\right)-\sqrt{\mu_m/r_f}+\omega r_f \end{bmatrix} \tag{3.156}$$

where $\tilde{r}=\sqrt{r_c^2\left(1\right)+d^2-2r_c\left(1\right)d\cos\theta_c\left(1\right)}$.

The control parameters, α_e, α_c, and α_m, are the time-of-flights of the Earth departure, translunar coast, and Moon capture, respectively. Equations (3.149) are the three-body thrusting equations in an Earth centered frame, Equations (3.150) are the three-body non-thrusting equations in an Earth centered frame, and Equations (3.151) are the three-body thrusting equations in a Moon centered reference frame. The constraints are grouped into four terms which represent the respective "matching" of the three separate trajectory phases as well as the initial and terminal conditions:

1. ψ_o are the required initial conditions in LEO.

2. $\psi_{e/c}$ matches the final conditions of the Earth departure to the initial conditions of the translunar coast.

3. $\psi_{c/m}$ matches the final conditions of the translunar coast to the initial conditions of the Lunar capture. This constraint vector utilizes the transformation between an Earth-centered to a Lunar-centered reference frame described in the previous section.

4. ψ_f are the required final conditions in LLO.

The derivation of the 2PBVP will be facilitated if the following definitions are made:

$$\bar{r} = [\bar{r}_e \;\; \bar{r}_c \;\; \bar{r}_m]^T \tag{3.157}$$

$$F = [F_e \;\; F_c \;\; F_m]^T \tag{3.158}$$

$$u = [u_e \;\; u_m]^T \tag{3.159}$$

The Hamiltonian can then be defined as

$$H = \lambda^T F \tag{3.160}$$

so that the costate equations are

$$\dot{\lambda} = -\left(\frac{\partial F}{\partial \bar{r}}\right)^T \lambda \tag{3.161}$$

$$= \begin{bmatrix} \alpha_e \left(\frac{\partial f_e}{\partial \bar{r}_e}\right)^T & 0 & 0 \\ 0 & \alpha_c \left(\frac{\partial f_c}{\partial \bar{r}_c}\right)^T & 0 \\ 0 & 0 & \alpha_m \left(\frac{\partial f_m}{\partial \bar{r}_m}\right)^T \end{bmatrix} \lambda \tag{3.162}$$

These equations result in identical costate equations as given by Equations (3.110) - (3.134) and will not be repeated here. The stationarity conditions yields the optimal control:

$$H_u = 0 = \left(\frac{\partial F}{\partial u}\right)^T \lambda \tag{3.163}$$

$$= \begin{bmatrix} \alpha_e \left(\frac{\partial f_e}{\partial u_e}\right)^T & 0 & 0 \\ 0 & 0 & \alpha_e \left(\frac{\partial f_m}{\partial u_m}\right)^T \end{bmatrix} \lambda \tag{3.164}$$

The transversality condition for a problem with initial and terminal state constraints and control parameters, see Appendix C, is given by:

$$\left(\phi_{\bar{r}} + \Psi_{\bar{r}}^T \nu - \lambda\right)_{\tau=1}^T \delta\bar{r}(1) + \left(\Psi_{\bar{r}}^T \nu + \lambda\right)_{\tau=0}^T \delta\bar{r}(0) + \left(\phi_\alpha + \int_0^1 H_\alpha d\tau\right) \delta\alpha = 0 \tag{3.165}$$

Since the three terms are independent, the resulting boundary conditions on the costates are

$$\lambda(1) = \phi_{\bar{r}} + \Psi_{\bar{r}(1)}^T \nu \tag{3.166}$$

$$\lambda(0) = -\Psi_{\bar{r}(0)}^T \nu \tag{3.167}$$

and the final term yields

$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \int_0^1 \begin{bmatrix} f_e & 0 & 0 \\ 0 & f_c & 0 \\ 0 & 0 & f_m \end{bmatrix}^T \lambda d\,\tau \tag{3.168}$$

Two constraint jacobians are needed to determine the costate boundary conditions. The first which will be found is for the initial conditions and is given by

$$\frac{\partial \Psi}{\partial \bar{r}(0)} = \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e(0)} & \frac{\partial \psi_o}{\partial \bar{r}_c(0)} & \frac{\partial \psi_o}{\partial \bar{r}_m(0)} \\ \frac{\partial \psi_{e/c}}{\partial \bar{r}_e(0)} & \frac{\partial \psi_{e/c}}{\partial \bar{r}_c(0)} & \frac{\partial \psi_{e/c}}{\partial \bar{r}_m(0)} \\ \frac{\partial \psi_{c/m}}{\partial \bar{r}_e(0)} & \frac{\partial \psi_{c/m}}{\partial \bar{r}_c(0)} & \frac{\partial \psi_{c/m}}{\partial \bar{r}_m(0)} \\ \frac{\partial \psi_f}{\partial \bar{r}_e(0)} & \frac{\partial \psi_f}{\partial \bar{r}_c(0)} & \frac{\partial \psi_f}{\partial \bar{r}_m(0)} \end{bmatrix} \tag{3.169}$$

$$= \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e(0)} & 0 & 0 \\ 0 & \frac{\partial \psi_{e/c}}{\partial \bar{r}_c(0)} & 0 \\ 0 & 0 & \frac{\partial \psi_{c/m}}{\partial \bar{r}_m(0)} \\ 0 & 0 & 0 \end{bmatrix} \tag{3.170}$$

where

$$\frac{\partial \psi_o}{\partial \bar{r}_e(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.171}$$

$$\frac{\partial \psi_{e/c}}{\partial \bar{r}_c(0)} = \mathrm{I} \tag{3.172}$$

$$\frac{\partial \psi_{c/m}}{\partial \bar{r}_m(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\cos\theta_m(0)}{r_c(1)} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.173}$$

which results in an initial costate vector of

$$\lambda_1(0) = -\nu_1 \tag{3.174}$$

$$\lambda_2(0) = 0 \tag{3.175}$$

$$\lambda_3(0) = -\nu_2 \tag{3.176}$$

$$\lambda_4(0) = -\nu_3 \tag{3.177}$$

$$\lambda_5(0) = -\nu_4 \tag{3.178}$$

$$\lambda_6(0) = -\nu_5 \tag{3.179}$$

$$\lambda_7(0) = -\nu_6 \tag{3.180}$$

$$\lambda_8(0) = -\nu_7 \tag{3.181}$$

$$\lambda_9(0) = -\nu_8 \tag{3.182}$$

$$\lambda_{10}(0) = -\frac{\cos\theta_m(0)}{r_c(1)}\nu_9 \tag{3.183}$$

$$\lambda_{11}(0) = -\nu_{10} \tag{3.184}$$

$$\lambda_{12}(0) = -\nu_{11} \tag{3.185}$$

The second Jacobian for the costate boundary conditions is

$$\frac{\partial \Psi}{\partial \bar{r}\,(1)} = \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e(1)} & \frac{\partial \psi_o}{\partial \bar{r}_c(1)} & \frac{\partial \psi_o}{\partial \bar{r}_m(1)} \\ \frac{\partial \psi_{e/c}}{\partial \bar{r}_e(1)} & \frac{\partial \psi_{e/c}}{\partial \bar{r}_c(1)} & \frac{\partial \psi_{e/c}}{\partial \bar{r}_m(1)} \\ \frac{\partial \psi_{c/m}}{\partial \bar{r}_e(1)} & \frac{\partial \psi_{c/m}}{\partial \bar{r}_c(1)} & \frac{\partial \psi_{c/m}}{\partial \bar{r}_m(1)} \\ \frac{\partial \psi_f}{\partial \bar{r}_e(1)} & \frac{\partial \psi_f}{\partial \bar{r}_c(1)} & \frac{\partial \psi_f}{\partial \bar{r}_m(1)} \end{bmatrix} \tag{3.186}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ \frac{\partial \psi_{e/c}}{\partial \bar{r}_e(1)} & 0 & 0 \\ 0 & \frac{\partial \psi_{c/m}}{\partial \bar{r}_c(1)} & 0 \\ 0 & 0 & \frac{\partial \psi_f}{\partial \bar{r}_m(1)} \end{bmatrix} \tag{3.187}$$

where

$$\frac{\partial \psi_{e/c}}{\partial \bar{r}_e\,(1)} = -\mathrm{I} \tag{3.188}$$

$$\frac{\partial \psi_{c/m}}{\partial \bar{r}_c\,(1)} = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \tag{3.189}$$

$$\frac{\partial \psi_f}{\partial \bar{r}_m\,(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.190}$$

and where

$$A_{11} = \frac{d\cos\theta_c - r_c}{\tilde{r}} \tag{3.191}$$

$$A_{21} = -\frac{\sin\theta_m\,(0)}{r_c^2} - \frac{\sin\theta_c\,[d\cos\theta_c - r_c]}{\tilde{r}^3} \tag{3.192}$$

$$A_{31} = \frac{V_{r_c}\cos\theta_c}{\tilde{r}} + \frac{(r_c - d\cos\theta_c)\,[(d - r_c\cos\theta_c)\,V_{r_c} - d\sin\theta_c V_{\theta_c}]}{\tilde{r}^3} \tag{3.193}$$

$$A_{41} = \frac{V_{\theta_c}\cos\theta_c}{\tilde{r}} \tag{3.194}$$

$$+\frac{(r_c - d\cos\theta_c)\left[d\sin\theta_c V_{r_c} + (d - r_c\cos\theta_c)\, V_{\theta_c}\right]}{\tilde{r}^3}$$

$$A_{12} = \frac{r_c d\sin\theta_c}{\tilde{r}} \tag{3.195}$$

$$A_{22} = \frac{r_c d\sin^2\theta_c}{\tilde{r}^3} - \frac{\cos\theta_c}{\tilde{r}} \tag{3.196}$$

$$A_{32} = \frac{dV_{\theta_c}\cos\theta_c - r_c V_{r_c}\sin\theta_c}{\tilde{r}} \tag{3.197}$$

$$+\frac{r_c d\sin\theta_c\left[(d - r_c\cos\theta_c)\, V_{r_c} - d\sin\theta_c V_{\theta_c}\right]}{\tilde{r}^3}$$

$$A_{42} = \frac{-r_c V_{\theta_c}\sin\theta_c - dV_{r_c}\cos\theta_c}{\tilde{r}} \tag{3.198}$$

$$+\frac{r_c d\sin\theta_c\left[dV_{r_c}\sin\theta_c + (d - r_c\cos\theta_c)\, V_{\theta_c}\right]}{\tilde{r}^3}$$

$$A_{33} = \frac{d - r_c\cos\theta_c}{\tilde{r}} \tag{3.199}$$

$$A_{43} = \frac{d\sin\theta_c}{\tilde{r}} \tag{3.200}$$

$$A_{34} = -\frac{d\sin\theta_c}{\tilde{r}} \tag{3.201}$$

$$A_{44} = \frac{d - r_c\cos\theta_c}{\tilde{r}} \tag{3.202}$$

where all of the coast variables of A_{11} - A_{44} are evaluated at $\tau = 1$. This Jacobian results in a final costate vector of

$$\lambda_1(1) = -\nu_4 \tag{3.203}$$

$$\lambda_2(1) = -\nu_5 \tag{3.204}$$

$$\lambda_3(1) = -\nu_6 \tag{3.205}$$

$$\lambda_4(1) = -\nu_7 \tag{3.206}$$

$$\lambda_5(1) = A_{11}\nu_8 + A_{21}\nu_9 + A_{31}\nu_{10} + A_{41}\nu_{11} \tag{3.207}$$

$$\lambda_6(1) = A_{12}\nu_8 + A_{22}\nu_9 + A_{32}\nu_{10} + A_{42}\nu_{11} \tag{3.208}$$

$$\lambda_7(1) = A_{33}\nu_{10} + A_{43}\nu_{11} \tag{3.209}$$

$$\lambda_8(1) = A_{34}\nu_{10} + A_{44}\nu_{11} \tag{3.210}$$

$$\lambda_9(1) = -\nu_{12} \tag{3.211}$$

$$\lambda_{10}(1) = 0 \tag{3.212}$$

$$\lambda_{11}(1) = \nu_{13} \tag{3.213}$$

$$\lambda_{12}(1) = \nu_{14} \tag{3.214}$$

Four of the constant ν_i's can be eliminated immediately by noting that

$$\lambda_5(0) = \lambda_1(1) \tag{3.215}$$

$$\lambda_6(0) = \lambda_2(1) \tag{3.216}$$

$$\lambda_7(0) = \lambda_3(1) \tag{3.217}$$

$$\lambda_8(0) = \lambda_4(1) \tag{3.218}$$

Another four can be eliminated by noticing the relationships between ν_9 - ν_{12} for both the initial and final times:

$$\begin{bmatrix} \lambda_9(0) \\ \lambda_{10}(0) \\ \lambda_{11}(0) \\ \lambda_{12}(0) \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & \frac{\cos\theta_m(0)}{r_c(1)} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \nu_8 \\ \nu_9 \\ \nu_{10} \\ \nu_{11} \end{bmatrix} \tag{3.219}$$

$$\begin{bmatrix} \lambda_5(1) \\ \lambda_6(1) \\ \lambda_7(1) \\ \lambda_8(1) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ 0 & 0 & A_{33} & A_{43} \\ 0 & 0 & A_{34} & A_{44} \end{bmatrix} \begin{bmatrix} \nu_8 \\ \nu_9 \\ \nu_{10} \\ \nu_{11} \end{bmatrix} \tag{3.220}$$

Solving for the unknown constants, ν's, gives the relationship between the costate boundary conditions:

$$\begin{bmatrix} \lambda_5(1) \\ \lambda_6(1) \\ \lambda_7(1) \\ \lambda_8(1) \end{bmatrix} = \begin{bmatrix} -A_{11} & -\frac{\cos\theta_m(0)A_{21}}{r_c(1)} & -A_{31} & -A_{41} \\ -A_{12} & -\frac{\cos\theta_m(0)A_{22}}{r_c(1)} & -A_{32} & -A_{42} \\ 0 & 0 & -A_{33} & -A_{43} \\ 0 & 0 & -A_{34} & -A_{44} \end{bmatrix} \begin{bmatrix} \lambda_9(0) \\ \lambda_{10}(0) \\ \lambda_{11}(0) \\ \lambda_{12}(0) \end{bmatrix} \tag{3.221}$$

The boundary conditions for the 2PBVP is then completely defined by Equations (3.175), (3.212), (3.215) - (3.218), (3.221), and the constraints (3.152). Three more conditions

are required because of the three control parameters, i.e., the times-of-flight, and these conditions come from Equation (3.168).

The benefit of hindsight allows a more aesthetic formulation. The costate vector, λ, can be separated into three components:

$$\lambda = \begin{bmatrix} \lambda_e \\ \lambda_c \\ \lambda_m \end{bmatrix} \tag{3.222}$$

which allows the costate differential equations to be written as

$$\dot{\lambda}_e = -\alpha_e \frac{\partial f_e}{\partial \bar{r}_e} \lambda_e \tag{3.223}$$

$$\dot{\lambda}_c = -\alpha_c \frac{\partial f_c}{\partial \bar{r}_c} \lambda_c \tag{3.224}$$

$$\dot{\lambda}_m = -\alpha_m \frac{\partial f_m}{\partial \bar{r}_m} \lambda_m \tag{3.225}$$

The stationarity condition yields the familiar tangent control law,

$$u_e^* = \tan^{-1}\left(\frac{-\lambda_{e_{V_r}}}{-\lambda_{e_{V_\theta}}}\right) \tag{3.226}$$

$$u_m^* = \tan^{-1}\left(\frac{-\lambda_{m_{V_r}}}{-\lambda_{m_{V_\theta}}}\right) \tag{3.227}$$

The greatest simplification comes in the transversality conditions,

$$\lambda_c(0) = \lambda_e(1) \tag{3.228}$$

$$\lambda_c(0) = A\lambda_m(0) \tag{3.229}$$

where

$$A = \begin{bmatrix} -A_{11} & -\frac{r_c(1)A_{21}}{\cos\theta_m(0)} & -A_{31} & -A_{41} \\ -A_{12} & -\frac{r_c(1)A_{22}}{\cos\theta_m(0)} & -A_{32} & -A_{42} \\ 0 & 0 & -A_{33} & -A_{43} \\ 0 & 0 & -A_{34} & -A_{44} \end{bmatrix} \tag{3.230}$$

and the individual elements are the same as previously written. The three control parameters, or times-of-flight, yield

$$1 + \int_0^1 f_e^T \lambda_e d\tau = 0 \tag{3.231}$$

$$\int_0^1 f_c^T \lambda_c d\tau = 0 \tag{3.232}$$

$$1 + \int_0^1 f_m^T \lambda_m d\tau = 0 \tag{3.233}$$

The solution of the 2PBVP is accomplished by enforcing the boundary conditions as described earlier in this chapter. The initial estimate for the solution of the 2PBVP is the solution returned by the hybrid "direct/indirect" method of the previous section. A terminal error function approach is can be used because of the accurate initial estimate which the hybrid "direct/indirect" returns. The NLP design variables are the thirty initial states and costates and three times-of-flight. Two different objective function and constraint vector configurations were investigated. The first configuration uses Equation (3.232) in the NLP objective function:

$$F = \left[\int_0^1 f_c^T \lambda_c d\tau\right]^2 \tag{3.234}$$

where the solution to the 2PBVP will give $\min F = 0$. The first configuration directly enforces the rest of the thirty-two conditions above in the NLP program. All of the solutions found from the hybrid method converged using this configuration where the smallest number of required iterations was 23, the largest was 112, and the average was 34.6. The second configuration does not impose any of the constraints directly in the NLP algorithm but uses a weighted constraint violation as the NLP objective function,

$$F = c^T W c \tag{3.235}$$

where W is positive definite and can be weighted towards those portions of c which are either more sensitive or have a greater initial violation. The vector c is the boundary and control parameter conditions in a vector format. The solution of this configuration

with a zero objective function solves the 2PBVP as long as W is positive definite. The simplest choice of W is simply the identity matrix which was used for these results. All of the solutions converged for this configuration. The smallest, largest, and average number of iterations required were 32, 67, and 36.1, respectively. There is essentially no difference between the two configurations with respect to convergence time or properties, and the ease with which either scheme solves the 2PBVP further shows the accuracy of the initial estimates.

CHAPTER 4 LUNAR TRANSFERS WITH SWITCHING

The optimal transfers described in detail in Chapter 3 all assumed a fixed engine firing sequence of thrust-coast-thrust. The actual optimal firing sequence for a power limited spacecraft will be governed by a switching function using both state and costate information during the trajectory. The solution of this problem with a full switching function in the Earth-Moon system is extremely difficult to find and the shooting/terminal error method of Chapter 3 fails to provide a practical approach. Switching transfers for low-thrust optimal Lunar transfers were investigated in Kluever and Pierson [30]. However, the boundary conditions they used were altered and did not simulate a complete transfer from LEO to LLO. A new method relying on discretization and using MINLP to approximate the switching structure is presented in this chapter. The process by which the true solution can be obtained from this MINLP approximation in a straight forward and repeatable manner is also presented. Finally, a series of numerical results are given for a specific mission type.

Classical Switching Structure

The problem formulation of the optimal transfer with a power limited vehicle requires some modifications from the problem statements presented in Chapter 3. The thrust of the spacecraft is given by

$$T = c\beta \tag{4.1}$$

where c is the effective exhaust velocity of the propellant and β is the mass flow. The exhaust velocity is assumed to be constant, so the thrust can be varied by changing the mass flow between zero and some upper maximum, β_{max}. The mass flow becomes a new control with an upper and lower bound, i.e., $0 \leq \beta(t) \leq \beta_{max}$. The previous planar transfers used three-body dynamics for both an Earth-centered and a Lunar-centered coordinate system. This was necessary because of the difficulty in accurately approximating the dynamic equations in an Earth-centered system for that portion of the trajectory close to the Moon. The inclusion of switching only increases this difficulty and both sets of dynamics are still required. However, the previous use of the two systems also indicated when thrusting would be turned off or on. This is not the case for the following problem formulation, and the change in coordinate systems is governed by the radial distance to the Moon and not by when the thrust is initiated.

The flight-times for the optimal thrust-coast-thrust transfers of the preceeding chapter were determined by the optimal process. The flight time for a switching transfer is necessarily fixed because a free-flight time switching transfer has an infinite time solution where the engine would be fired for an infinitesimal time and the spacecraft would take an infinite amount of time to make the transfer. A specific switching structure is dependent upon the specified time-of-flight where its logical that as the time-of-flight is increased the number of switches is increased to allow a more efficient transfer. The general problem statement for the planar switching problem is:

Find the geocentric thrust steering angle $u_e(\tau)$, geocentric mass flow $\beta_e(\tau)$, lunar thrust steering angle $u_m(\tau)$, and lunar mass flow $\beta_m(\tau)$ during the time interval $0 \leq \tau \leq 1$ which minimizes:

$$\phi[\bar{r}(t_f), t_f] = -m_m(1) \tag{4.2}$$

subject to

$$\frac{d\bar{r}_e}{d\tau} = \alpha_e f_e(\bar{r}_e, \bar{u}_e) \tag{4.3}$$

$$\frac{d\bar{r}_m}{d\tau} = \alpha_m f_m(\bar{r}_m, \bar{u}_m) \tag{4.4}$$

where

$$\bar{r}_e = [r_e \;\; \theta_e \;\; V_{r_e} \;\; V_{\theta_e} \;\; m_e]^T \tag{4.5}$$

$$\bar{u}_e = [u_e \;\; \beta_e]^T \tag{4.6}$$

$$\bar{r}_m = [r_m \;\; \theta_m \;\; V_{r_m} \;\; V_{\theta_m} \;\; m_m]^T \tag{4.7}$$

$$\bar{u}_m = [u_m \;\; \beta_m]^T \tag{4.8}$$

$$\tag{4.9}$$

and subject to the boundary conditions

$$\Psi[\bar{r}(0), \bar{r}(1)] = \begin{bmatrix} \psi_o \\ \psi_{e/m} \\ \psi_f \end{bmatrix} = 0 \tag{4.10}$$

where

$$\psi_o = \begin{bmatrix} r_e(0) - r_{LEO} \\ V_{r_e}(0) \\ V_{\theta_e}(0) - \sqrt{\mu_e / r_{LEO}} + \omega r_{LEO} \\ m_e(0) - m_{LEO} \end{bmatrix} \tag{4.11}$$

$$\psi_{e/m} = \begin{bmatrix} r_m(0) - \tilde{r}(r_e(1), \theta_e(1)) \\ \sin(\theta_m(0))/r_e(1) - \sin(\theta_e(1))/\tilde{r}(r_e(1), \theta_e(1)) \\ V_{r_m}(0) - [(d - r_e(1)\cos\theta_e(1)) V_{r_e}(1) - d\sin\theta_e(1) V_{\theta_e}(1)]/\tilde{r}(r_e(1), \theta_e(1)) \\ V_{\theta_m}(0) + [d\sin\theta_e(1) V_{r_e}(1) - (d - r_e(1)\cos\theta_e(1)) V_{\theta_e}(1)]/\tilde{r}(r_e(1), \theta_e(1)) \\ m_m(0) = m_e(1) \end{bmatrix} \tag{4.12}$$

$$\psi_f = \begin{bmatrix} r_m(1) - r_{LLO} \\ V_{r_m}(1) \\ V_{\theta_m}(1) \pm \sqrt{\mu_m / r_{LLO}} + \omega r_{LLO} \end{bmatrix} \tag{4.13}$$

where $\tilde{r} = \tilde{r}(r, \theta) = \sqrt{r^2 + d^2 - 2rd\cos\theta}$. The two mass flow controls are bounded by $0 \leq \beta_e \leq \beta_{max}$ and $0 \leq \beta_m \leq \beta_{max}$. Equation (4.3) represents the three-body equations (3.68) - (3.72) in an Earth-centered frame and Equation (4.4) represents the three-body equations (3.78) - (3.82) in a Lunar-centered frame. The two parameters α_e and α_m are the times spent in the Earth-centered frame and Lunar-centered frame, respectively, and the total time-of-flight is then the sum of the two. Note that both α's are fixed for this problem; they are not control parameters as they were for the 2PBVP formulation in the final section of Chapter 3. Because the α's are not control parameters and are chosen just to determine the time of the coordinate transformation, multiple burn and coast arcs can occur during each time-of-flight, i.e., during the time spent in the Earth-centered equations α_e a number of switches can occur and the same can be said for the time spent in the Lunar-centered equations α_m. Equation (4.11) specifies the initial circular LEO, Equation (4.12) links the two reference frame boundaries together, and Equation (4.13) specifies the final circular LLO.

The formulation of the optimal control problem is easier with the following definitions:

$$F \doteq \begin{bmatrix} \alpha_e f_e \\ \alpha_m f_m \end{bmatrix} \tag{4.14}$$

$$U \doteq \begin{bmatrix} \bar{u}_e \\ \bar{u}_m \end{bmatrix} \tag{4.15}$$

The Hamiltonian is then

$$H = L + \lambda^T F = \lambda^T F \tag{4.16}$$

and defining the costate vector, λ, in the same format as the last section of Chapter 3, i.e.,

$$\lambda = \begin{bmatrix} \lambda_e \\ \lambda_m \end{bmatrix} \tag{4.17}$$

allows the costate equations to be written as

$$\frac{\partial F}{\partial \bar{r}} = \begin{bmatrix} \alpha_e \frac{\partial f_e}{\partial \bar{r}_e} & 0 \\ 0 & \alpha_m \frac{\partial f_m}{\partial \bar{r}_m} \end{bmatrix} \tag{4.18}$$

$$\rightarrow \dot{\lambda}_e = -\alpha_e \left(\frac{\partial f_e}{\partial \bar{r}_e} \right)^T \lambda_e \tag{4.19}$$

$$\dot{\lambda}_m = -\alpha_m \left(\frac{\partial f_m}{\partial \bar{r}_m} \right)^T \lambda_m \tag{4.20}$$

These equations are identical to the costate equations described for the thrusting portions of the problems in Chapter 3, i.e., Equations (3.110) - (3.113) and Equations (3.131) - (3.134). The stationarity condition yields the optimal control:

$$H_u = 0 = \left(\frac{\partial F}{\partial U} \right)^T \lambda \tag{4.21}$$

and splitting the function and control vectors into their components yields,

$$0 = \begin{bmatrix} \alpha_e \left(\frac{\partial f_e}{\partial u_e} \right)^T \lambda_e \\ \alpha_e \left(\frac{\partial f_e}{\partial \beta_e} \right)^T \lambda_e \\ \alpha_m \left(\frac{\partial f_m}{\partial u_m} \right)^T \lambda_m \\ \alpha_m \left(\frac{\partial f_m}{\partial \beta_m} \right)^T \lambda_m \end{bmatrix} \tag{4.22}$$

The first and third terms yield the tangent steering laws:

$$\tan u_e^* = \frac{\lambda_{V_{r_e}}}{\lambda_{V_{\theta_e}}} \tag{4.23}$$

$$\tan u_m^* = \frac{\lambda_{V_{r_m}}}{\lambda_{V_{\theta_m}}} \tag{4.24}$$

where the sign of the separate trigonometric relations again need to be determined by the Legendre-Clebsch condition. The Hessian is:

$$H_{uu} = \begin{bmatrix} H_{uu}^{(e)} & 0 \\ 0 & H_{uu}^{(m)} \end{bmatrix} \tag{4.25}$$

where

$$H_{uu}^{(e)} = \frac{\alpha_e c}{m_e}\begin{bmatrix} -\beta_e\left(\lambda_{V_{r_e}}\sin u_e + \lambda_{V_{\theta_e}}\cos u_e\right) & \lambda_{V_{r_e}}\cos u_e - \lambda_{V_{\theta_e}}\sin u_e \\ \lambda_{V_{r_e}}\cos u_e - \lambda_{V_{\theta_e}}\sin u_e & 0 \end{bmatrix} \tag{4.26}$$

$$H_{uu}^{(m)} = \frac{\alpha_m c}{m_m}\begin{bmatrix} -\beta_m\left(\lambda_{V_{r_m}}\sin u_m + \lambda_{V_{\theta_m}}\cos u_m\right) & \lambda_{V_{r_m}}\cos u_m - \lambda_{V_{\theta_m}}\sin u_m \\ \lambda_{V_{r_m}}\cos u_m - \lambda_{V_{\theta_m}}\sin u_m & 0 \end{bmatrix} \tag{4.27}$$

The eigenvalues of H_{uu} are

$$\det\left(\phi I - H_{uu}\right) = \det\left(\phi I - H_{uu}^{(e)}\right)\det\left(\phi I - H_{uu}^{(e)}\right) \tag{4.28}$$

Note that since the sign of $\sin u_i$ and $\cos u_i$ are the same, the off-diagonal terms in both blocks are zero. The eigenvalues then become:

$$\phi_{1,2} = 0 \tag{4.29}$$

$$\phi_3 = -\frac{\alpha_e c\beta_e}{m_e}\left(\lambda_{V_{r_e}}\sin u_e + \lambda_{V_{\theta_e}}\cos u_e\right) \tag{4.30}$$

$$\phi_4 = -\frac{\alpha_m c\beta_m}{m_m}\left(\lambda_{V_{r_m}}\sin u_m + \lambda_{V_{\theta_m}}\cos u_m\right) \tag{4.31}$$

Requiring $H_{uu} \geq 0$ yields the negative sign for the trigonometric functions,

$$\sin u_e^* = \frac{-\lambda_{V_{r_e}}}{|\lambda_{V_e}|} \tag{4.32}$$

$$\cos u_e^* = \frac{-\lambda_{V_{\theta_e}}}{|\lambda_{V_e}|} \tag{4.33}$$

$$\sin u_m^* = \frac{-\lambda_{V_{r_m}}}{|\lambda_{V_m}|} \tag{4.34}$$

$$\cos u_m^* = \frac{-\lambda_{V_{\theta_m}}}{|\lambda_{V_m}|} \tag{4.35}$$

where $|\lambda_{V_e}| = \sqrt{\lambda_{V_{r_e}}^2 + \lambda_{V_{\theta_e}}^2}$ and $|\lambda_{V_m}| = \sqrt{\lambda_{V_{r_m}}^2 + \lambda_{V_{\theta_m}}^2}$.

The second and fourth term of Equation (4.22) are the switching functions which govern the massflow for both coordinate systems. The Earth-switching function is

$$\sigma_e = -\frac{c}{m_e}|\lambda_{V_e}| - \lambda_{m_e} \tag{4.36}$$

where the time-parameter α_e is dropped since it does not effect the sign of σ_e. The optimal geocentric massflow is

$$\beta_e^* = \begin{cases} \beta_{\max}, & \sigma_e < 0 \\ 0, & \sigma_e > 0 \end{cases} \tag{4.37}$$

The Lunar-switching function is

$$\sigma_m = -\frac{c}{m_m}|\lambda_{V_m}| - \lambda_{m_m} \tag{4.38}$$

and the optimal Lunar massflow is

$$\beta_m^* = \begin{cases} \beta_{\max}, & \sigma_m < 0 \\ 0, & \sigma_m > 0 \end{cases} \tag{4.39}$$

The transversality conditions for the initial and terminal state constraints (Appendix C) are

$$\left(\phi_{\bar{r}} + \Psi_{\bar{r}}^T \nu - \lambda\right)_{\tau=1}^T \delta\bar{r}(1) + \left(\Psi_{\bar{r}}^T \nu + \lambda\right)_{\tau=0}^T \delta\bar{r}(0) = 0 \tag{4.40}$$

Since the two terms are independent, the resulting boundary conditions on the costates are

$$\lambda(1) = \phi_{\bar{r}} + \Psi_{\bar{r}(1)}^T \nu \tag{4.41}$$

$$\lambda(0) = -\Psi_{\bar{r}(0)}^T \nu \tag{4.42}$$

Two constraint jacobians are needed to determine the costate boundary conditions. The first which will be found is for the initial conditions and is given by

$$\frac{\partial \Psi}{\partial \bar{r}(0)} = \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e} & \frac{\partial \psi_o}{\partial \bar{r}_m} \\ \frac{\partial \psi_{e/m}}{\partial \bar{r}_e} & \frac{\partial \psi_{e/m}}{\partial \bar{r}_m} \\ \frac{\partial \psi_f}{\partial \bar{r}_e} & \frac{\partial \psi_f}{\partial \bar{r}_m} \end{bmatrix}_{\tau=0} \tag{4.43}$$

$$= \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e} & 0 \\ 0 & \frac{\partial \psi_{e/m}}{\partial \bar{r}_c} \\ 0 & 0 \end{bmatrix}_{\tau=0} \tag{4.44}$$

where

$$\frac{\partial \psi_o}{\partial \bar{r}_e(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.45}$$

$$\frac{\partial \psi_{c/m}}{\partial \bar{r}_m(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{\cos\theta_m(0)}{r_e(1)} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.46}$$

which results in initial costate vectors of

$$\lambda_e(0) = [\, -\nu_1, \;\; 0, \;\; -\nu_2, \;\; -\nu_3, \;\; -\nu_4 \,]^T \tag{4.47}$$

$$\lambda_m(0) = \left[-\nu_5, \;\; -\frac{\cos\theta_m(0)}{r_e(1)}\nu_6, \;\; -\nu_7, \;\; -\nu_8, \;\; -\nu_9 \right]^T \tag{4.48}$$

The second jacobian for the costate boundary conditions is

$$\frac{\partial \Psi}{\partial \bar{r}(1)} = \begin{bmatrix} \frac{\partial \psi_o}{\partial \bar{r}_e} & \frac{\partial \psi_o}{\partial \bar{r}_m} \\ \frac{\partial \psi_{e/m}}{\partial \bar{r}_e} & \frac{\partial \psi_{e/m}}{\partial \bar{r}_m} \\ \frac{\partial \psi_f}{\partial \bar{r}_e} & \frac{\partial \psi_f}{\partial \bar{r}_m} \end{bmatrix}_{\tau=1} \tag{4.49}$$

$$= \begin{bmatrix} 0 & 0 \\ \frac{\partial \psi_{e/m}}{\partial \bar{r}_e} & 0 \\ 0 & \frac{\partial \psi_f}{\partial \bar{r}_m} \end{bmatrix}_{\tau=1} \tag{4.50}$$

where

$$\frac{\partial \psi_{e/m}}{\partial \bar{r}_e(1)} = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} & 0 \\ A_{41} & A_{42} & A_{43} & A_{44} & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{4.51}$$

$$\frac{\partial \psi_f}{\partial \bar{r}_m(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.52}$$

and where

$$A_{11} = \frac{d\cos\theta_e - r_e}{\tilde{r}} \tag{4.53}$$

$$A_{21} = -\frac{\sin\theta_m(0)}{r_e^2} - \frac{\sin\theta_e\left[d\cos\theta_e - r_e\right]}{\tilde{r}^3} \tag{4.54}$$

$$A_{31} = \frac{V_{r_e}\cos\theta_e}{\tilde{r}} + \frac{(r_e - d\cos\theta_e)\left[(d - r_e\cos\theta_e)V_{r_e} - d\sin\theta_e V_{\theta_e}\right]}{\tilde{r}^3} \tag{4.55}$$

$$A_{41} = \frac{V_{\theta_e}\cos\theta_e}{\tilde{r}} + \frac{(r_e - d\cos\theta_e)\left[d\sin\theta_e V_{r_e} + (d - r_e\cos\theta_e)V_{\theta_e}\right]}{\tilde{r}^3} \tag{4.56}$$

$$A_{12} = \frac{r_e d\sin\theta_e}{\tilde{r}} \tag{4.57}$$

$$A_{22} = \frac{r_e d\sin^2\theta_e}{\tilde{r}^3} - \frac{\cos\theta_e}{\tilde{r}} \tag{4.58}$$

$$A_{32} = \frac{dV_{\theta_e}\cos\theta_e - r_e V_{r_e}\sin\theta_e}{\tilde{r}} + \frac{r_e d\sin\theta_e\left[(d - r_e\cos\theta_e)V_{r_e} - d\sin\theta_e V_{\theta_e}\right]}{\tilde{r}^3} \tag{4.59}$$

$$A_{42} = \frac{-r_e V_{\theta_e}\sin\theta_e - dV_{r_e}\cos\theta_e}{\tilde{r}} + \frac{r_e d\sin\theta_e\left[dV_{r_e}\sin\theta_e + (d - r_e\cos\theta_e)V_{\theta_e}\right]}{\tilde{r}^3} \tag{4.60}$$

$$A_{33} = \frac{d - r_e\cos\theta_e}{\tilde{r}} \tag{4.61}$$

$$A_{43} = \frac{d\sin\theta_e}{\tilde{r}} \tag{4.62}$$

$$A_{34} = -\frac{d\sin\theta_e}{\tilde{r}} \tag{4.63}$$

$$A_{44} = \frac{d - r_e\cos\theta_e}{\tilde{r}} \tag{4.64}$$

where all of the Earth centered variables of A_{11} - A_{44} are evaluated at $\tau = 1$. This

jacobian results in a final costate vector of

$$\lambda_e(1) = \begin{bmatrix} A_{11}\nu_5 + A_{21}\nu_6 + A_{31}\nu_7 + A_{41}\nu_8 \\ A_{12}\nu_5 + A_{22}\nu_6 + A_{32}\nu_7 + A_{42}\nu_8 \\ A_{33}\nu_7 + A_{43}\nu_8 \\ A_{34}\nu_7 + A_{44}\nu_8 \\ -\nu_9 \end{bmatrix} \tag{4.65}$$

$$\lambda_m(1) = \begin{bmatrix} \nu_{10} \\ 0 \\ \nu_{11} \\ \nu_{12} \\ -1 \end{bmatrix} \tag{4.66}$$

The last term equating $\lambda_{m_e}(1)$ to -1 comes from the gradient of the performance index, ϕ_r. Three costate boundary values are immediately useful,

$$\lambda_{\theta_e}(0) = 0 \tag{4.67}$$

$$\lambda_{\theta_m}(1) = 0 \tag{4.68}$$

$$\lambda_{m_m}(1) = -1 \tag{4.69}$$

The last five needed conditions are obtained by relating ν_5 - ν_8:

$$\lambda_e(1) = \begin{bmatrix} -A_{11} & -\frac{r_e(1)}{\cos\theta_m(0)}A_{21} & -A_{31} & -A_{41} & 0 \\ -A_{12} & -\frac{r_e(1)}{\cos\theta_m(0)}A_{22} & -A_{32} & -A_{42} & 0 \\ 0 & 0 & -A_{33} & -A_{43} & 0 \\ 0 & 0 & -A_{34} & -A_{44} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \lambda_m(0) \tag{4.70}$$

The 2PBVP consists of the state equations, (4.3) and (4.4), and the costate equations, (4.19) and (4.20) with the boundary conditions of (4.11) - (4.13) and (4.67) - (4.70).

The time derivative of the switching function can be found and is useful later in this chapter. The subscripts, denoting which coordinate system the state and costate variables belong to, will be left off of the derivation since the derivation is independent of which system is used. By definition, the time derivative of a switching function is

$$\dot{\sigma} = \frac{d}{dt}\left(-\frac{c}{m}|\lambda_v| - \lambda_m\right) \tag{4.71}$$

$$= -c\frac{m\left|\dot{\lambda}_v\right| - |\lambda_v|\dot{m}}{m^2} - \dot{\lambda}_m \tag{4.72}$$

where $\left|\dot{\lambda}_v\right| = \left(\lambda_{V_R}\dot{\lambda}_{V_r} + \lambda_{V_\theta}\dot{\lambda}_{V_\theta}\right)/|\lambda_V|$ and

$$\dot{m} = -\alpha\beta \tag{4.73}$$

$$\dot{\lambda}_m = -\alpha\frac{c\beta}{m}|\lambda_V| \tag{4.74}$$

The derivative then becomes

$$\dot{\sigma} = -c\left(\frac{\lambda_{V_r}\dot{\lambda}_{V_r} + \lambda_{V_\theta}\dot{\lambda}_{V_\theta}}{m|\lambda_V|} + \frac{\alpha\beta}{m^2}|\lambda_V|\right) + \alpha\frac{c\beta}{m}|\lambda_V| \tag{4.75}$$

$$= -\frac{c}{m}\frac{\lambda_{V_r}\dot{\lambda}_{V_r} + \lambda_{V_\theta}\dot{\lambda}_{V_\theta}}{|\lambda_V|} \tag{4.76}$$

where

$$\dot{\lambda}_{V_r} = \alpha\left(-\lambda_r + \left(\frac{V_\theta}{r} + 2\omega\right)\lambda_{V_\theta}\right) \tag{4.77}$$

$$\dot{\lambda}_{V_\theta} = \alpha\left(-\frac{\lambda_\theta}{r} - 2\left(\frac{V_\theta}{r} + \omega\right)\lambda_{V_r} + \frac{V_r}{r}\lambda_{V_\theta}\right) \tag{4.78}$$

yields the derivative in terms of the current state and costate

$$\dot{\sigma} = -\frac{c}{m|\lambda_V|}\left(\frac{\lambda_{V_\theta}}{r}\left(V_r\lambda_{V_\theta} - V_\theta\lambda_{V_r} - \lambda_\theta\right) - \lambda_r\lambda_{V_r}\right) \tag{4.79}$$

The switching function is dependent upon the mass, the costate velocities, and the mass costate. The mass and the mass costate are identical between the coordinate systems as seen by Equation (4.12) and Equation (4.70). The velocity costates are also related

by the transversality conditions and that portion of the matrix of Equation (4.70) is repeated here

$$\lambda_{V_e}(1) = \begin{bmatrix} -A_{33} & -A_{43} \\ -A_{34} & -A_{44} \end{bmatrix} \lambda_{V_m}(0) \tag{4.80}$$

where

$$\lambda_{V_e} = \begin{bmatrix} \lambda_{V_{r_e}} \\ \lambda_{V_{\theta_e}} \end{bmatrix} \quad \text{and} \quad \lambda_{V_m} = \begin{bmatrix} \lambda_{V_{r_m}} \\ \lambda_{V_{\theta_m}} \end{bmatrix} \tag{4.81}$$

and where

$$A_{33} = \frac{d - r_e(1)\cos\theta_e(1)}{\tilde{r}} \tag{4.82}$$

$$A_{34} = \frac{d\sin\theta_e(1)}{\tilde{r}} \tag{4.83}$$

$$A_{43} = -A_{34} \tag{4.84}$$

$$A_{44} = A_{33} \tag{4.85}$$

$$\tilde{r} = \left(r_e^2(1) + d^2 - 2r_e(1)\,d\cos\theta_e(1)\right)^{1/2} \tag{4.86}$$

The square of the Earth velocity costates can then be written as

$$\lambda_{V_{r_e}}^2(1) = \frac{1}{\tilde{r}^2}\left[\lambda_{V_{r_m}}(0)\left(d - r_e(1)\cos\theta_e(1)\right) + \lambda_{V_{\theta_m}}(0)\,d\sin\theta_e(1)\right]^2 \tag{4.87}$$

$$\lambda_{V_{r_m}}^2(1) = \frac{1}{\tilde{r}^2}\left[-\lambda_{V_{r_m}}(0)\,d\sin\theta_e(1) + \lambda_{V_{\theta_m}}(0)\left(d - r_e(1)\cos\theta_e(1)\right)\right]^2 \tag{4.88}$$

Therefore, the velocity costate norm is

$$|\lambda_{V_e}(1)| = \left(\frac{d^2\sin^2\theta_e(1) + \left(d - r_e(1)\cos\theta_e(1)\right)^2}{\tilde{r}^2}\right)^{1/2} |\lambda_{V_m}(0)| \tag{4.89}$$

$$= \hat{T}\,|\lambda_{V_m}(0)| \tag{4.90}$$

The final condition of the Earth switching function can now be written as

$$\sigma_e(1) = -\frac{c}{m_e(1)}|\lambda_{V_e}(1)| - \lambda_{m_e}(1) \tag{4.91}$$

$$= -\frac{c}{m_m(0)}\hat{T}\,|\lambda_{V_m}(0)| - \lambda_{m_m}(0) \tag{4.92}$$

The initial value of the lunar switching function can then be written as

$$\sigma_m(0) = \sigma_e(1) + \frac{c}{m_e(1)}\left(1 - \hat{T}\right)|\lambda_{V_e}(1)| \tag{4.93}$$

Equation (4.93) gives the initial condition of the Lunar-centered switching function in terms of the final value of the Earth-centered switching function, the mass at the cross-over point, and the norm of the Earth-centered costate velocities. This equation becomes useful in the next section when the method by which a MINLP approximation is used to find the full solution of the 2PBVP.

An example solution shows typical characteristics of an optimal Lunar transfer switching trajectory. Figure 4.1 shows the posigrade trajectory for $(T/W)_i = 3(10^{-3})$ and a specified total flight time of 8.5 days. Figure 4.2 shows a close-up of the Lunar capture. The trajectory has a total of eight engine switches where three switches occur about the Earth and five about the Moon. The final mass to initial mass performance ratio is 0.939 which represents a 8.989 percent improvement over the fixed thrust-coast-thrust solution of Chapter 3 or a gain of 836.6 kg. The thrust steering control history for the Earth and Lunar centered portions of the trajectory are shown in Figures 4.3 and 4.4, respectively. The effects of switching on the control angles are immediately apparent. The thrusting periods are placed near the perigee points, and so the steering angle during those stages follows the flight path angle to increase the energy of the spacecraft in an optimal fashion. The switching function for the Earth-centered portion of the trajectory is shown in Figure 4.5 and the Lunar-centered switching function is shown in Figure 4.6. The oscillatory nature of the functions is expected since they are dependent upon the norm of the velocity costates. Note that small oscillations of the Earth switching functions occur during the long initial thrust phase. The energy of the spacecraft increases, along with the magnitude of σ_e, until enough energy is obtained and switching occurs, i.e., the engine is turned off, as indicated by the σ_e curve going positive. The coast phase and next orbit cause another switch followed by another

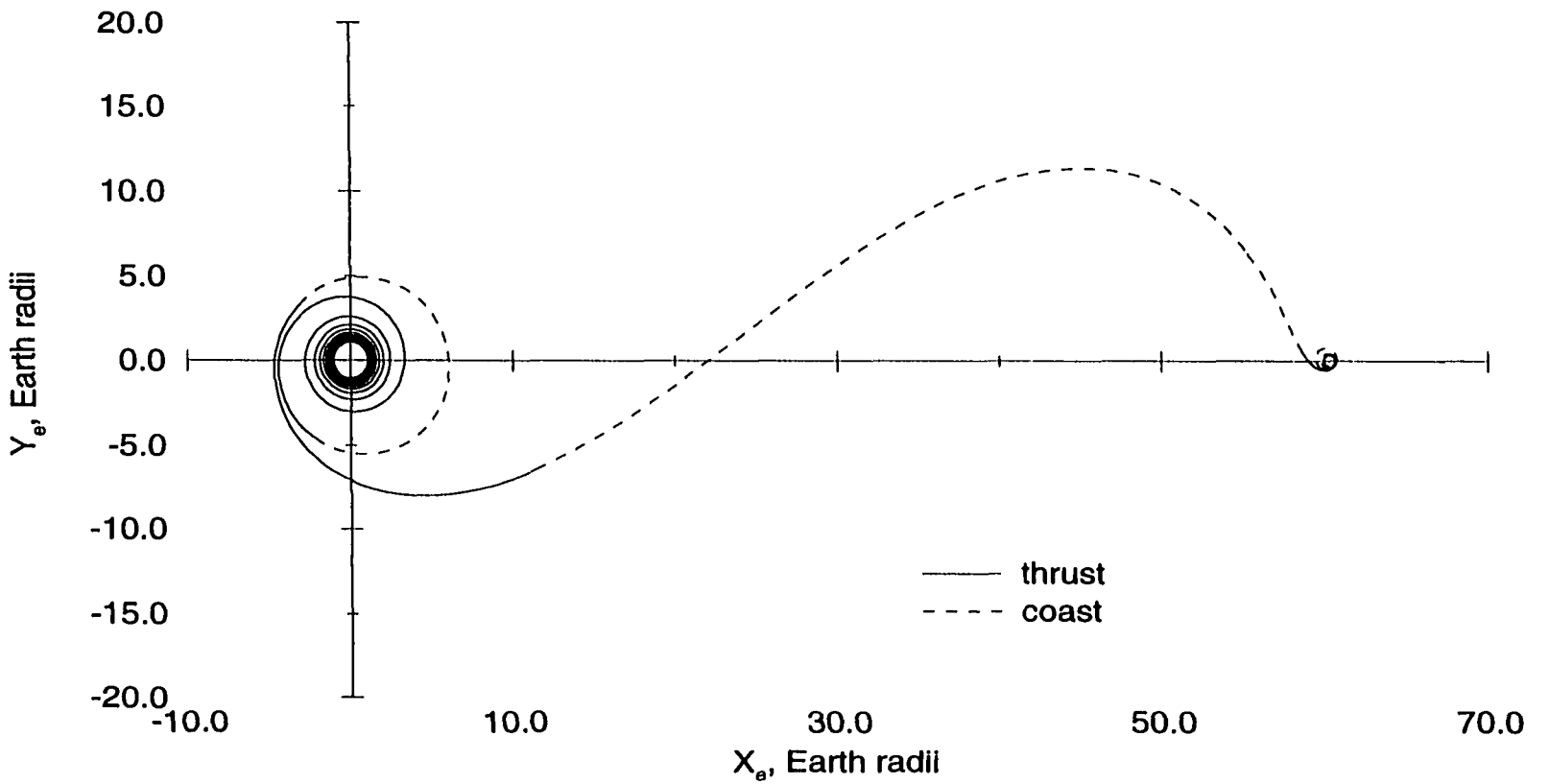

Figure 4.1 Posigrade Optimal Lunar Switching Transfer for a Flight Time of 8.5 days

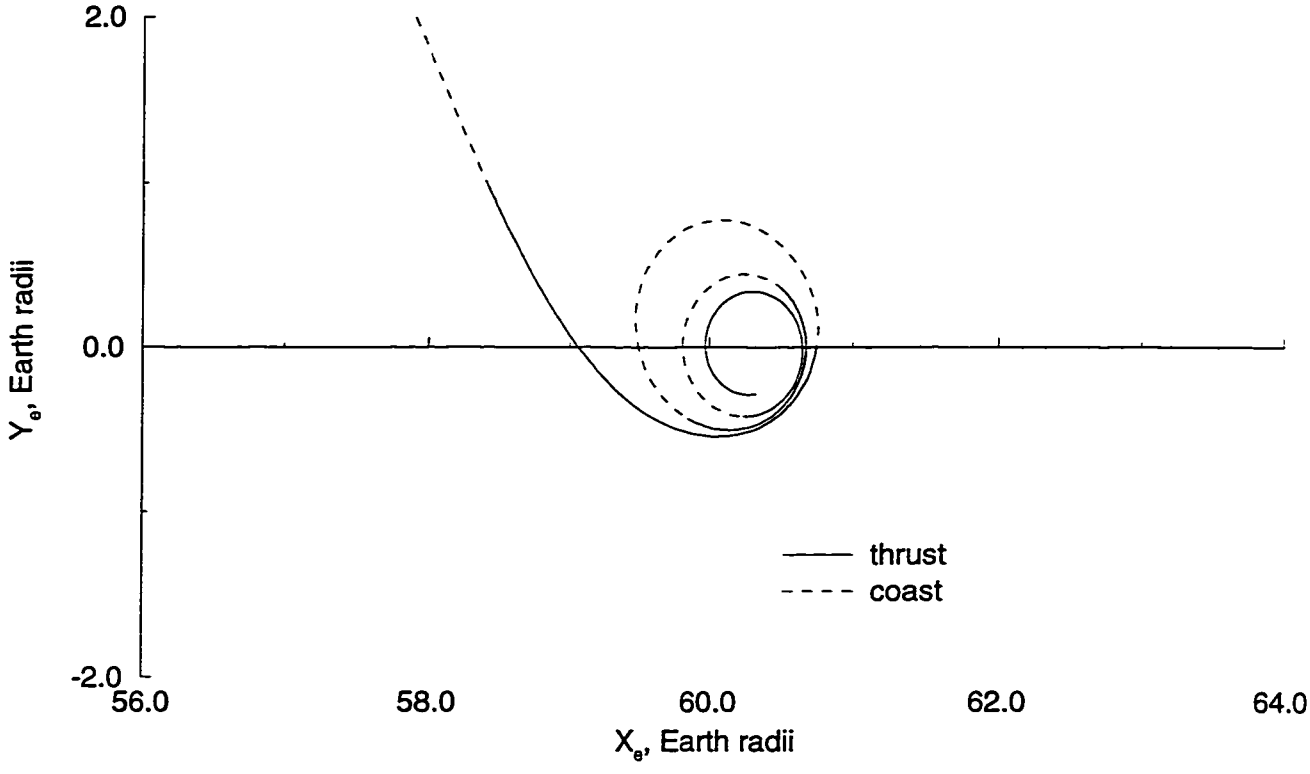

Figure 4.2 Posigrade Optimal Lunar Switching Transfer for a Flight Time of 8.5 days - Lunar Close-Up

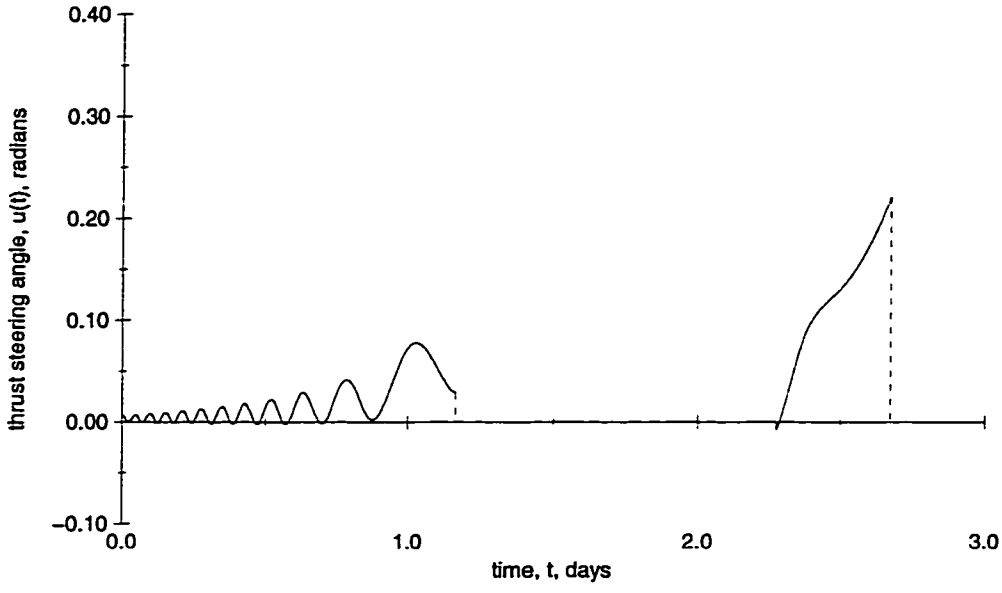

Figure 4.3 Earth-Centered Thrust Steering Angle for Posigrade Optimal Lunar Switching Transfer

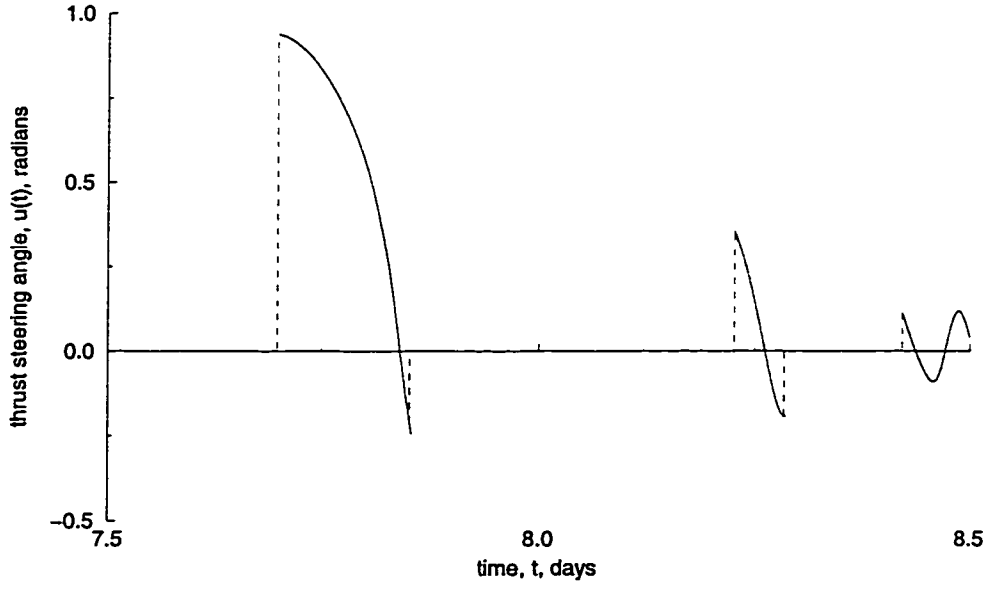

Figure 4.4 Lunar-Centered Thrust Steering Angle for Posigrade Optimal Lunar Switching Transfer

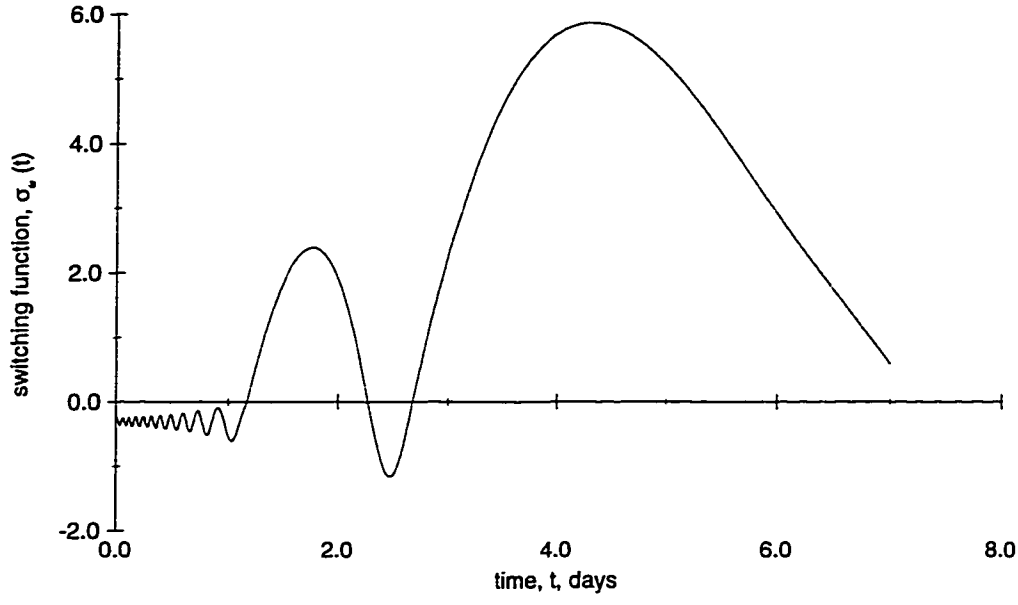

Figure 4.5 Earth-Centered Switching Function for Posigrade Optimal Lunar Transfer

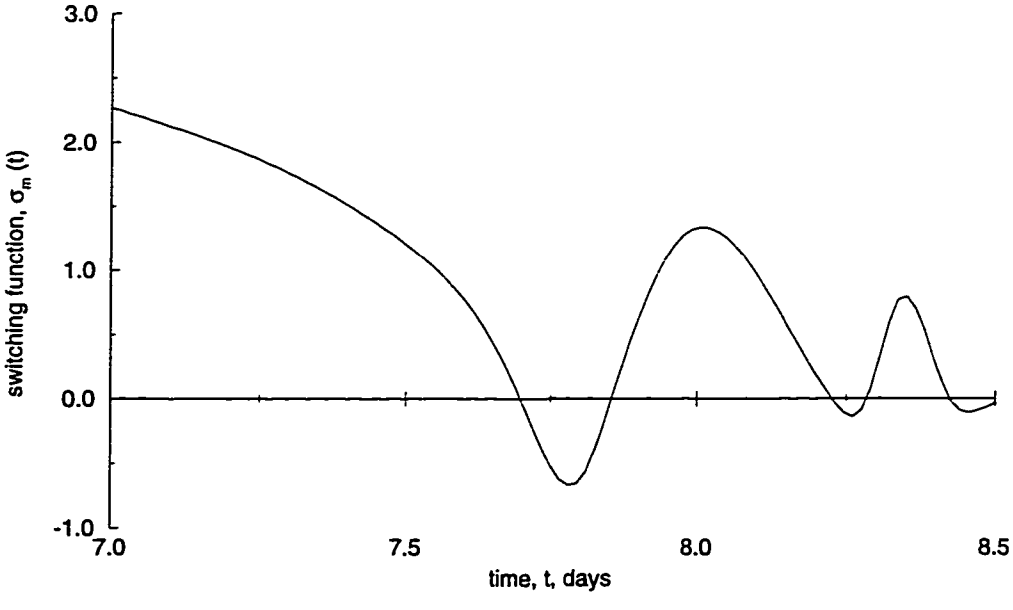

Figure 4.6 Lunar-Centered Switching Function for Posigrade Optimal Lunar Transfer

perigee burn which takes the spacecraft into the long translunar coast portion of the transfer. The lunar switching function σ_m does not have a period of small switching function oscillations as does σ_e, and instead switches with every oscillation which causes a five-switch capture.

MINLP Switching Approximation

The application of a MINLP algorithm to the solution of the switching problem requires a far different approach than the terminal error function described in Chapter 3. A different form of problem statement is described and the method of solution is presented in this section.

Background

Two main difficulties prevent the use of the terminal error function method for the solution of the switching problem. The first is the inability of most initial estimates to provide an effective means to allow a coast phase "inside" of a thrust phase. Intuitively, it's reasonable to expect the engine to fire at or near perigee in order to gain the most for the thrusting period. A coast phase would then begin, which also allows gravity assistance from the Moon to increase the spacecraft's energy, and this coast would end when the spacecraft is again at or near perigee. An NLP algorithm attempting to solve the 2PBVP from an initial estimate often introduces such a coast phase but usually does not allow the coast to extend for a long enough period of time before thrusting is re-initiated. Since the thrusting would then not occur at perigee, the resulting trial trajectory shows a performance loss and often results in a large constraint violation. Thus, the NLP algorithm is rarely able to introduce a new coasting phase and terminates with an error. This first difficulty could possibly be compensated for if the number and order of thrusting and coast periods could be determined for a specified time-of-

flight before the solution is attempted. The number of thrust and coast periods could be estimated for a time-of-flight and a related problem solved matching the boundary conditions. This solution could provide an accurate estimate for the unknown initial conditions so that the switching 2PBVP could be solved. However, it is not possible to know this firing sequence before hand and this prevents the practical use of this approach. The second difficulty is the inability to obtain a solution for the 2PBVP from a known previous solution. The optimal transfers in Chapter 3 provide a known solution of a thrust-coast-thrust sequence for a specified time-of-flight. The switching solution for an increased time-of-flight can not be found using this solution as an initial estimate. Both of these difficulties require a new method of approaching the problem.

The discretization/collocation method described in Chapter 2 also fails to solve the switching problem introduced in this chapter. All of the discretization and/or relaxation methods require gradient information along the trajectory. The mass flow is governed by a switching function and unless the function is very nearly zero, the gradient will return a zero and cause singularity problems in the solution of the constraint gradients.

MINLP Problem Setup

The discretization methods described in Chapter 2 present a method for using a NLP algorithm to find the discretized control which minimizes a given performance index. This idea is extended to include the switching function outlined in the previous section. Four control histories are required for the solution of a Lunar transfer. Two of these control histories, the thrust steering angles, are real variables and nothing new is introduced for their solution. The two mass flow controls are known to be discrete having a value of zero or some maximum value. The collocation methods can not solve for this switching function using a NLP algorithm, and a MINLP algorithm is used instead where the integer variables provide the switching approximation. A slight change in notation is introduced to prevent confusion between the state and control vectors at a point and

the state and controls at that point. The state vector at $t = t_i$ is denoted by $\bar{r}_i$ and the control vector is denoted by $\bar{u}_i$ where

$$\bar{r}_i = [\, r_i,\ \theta_i,\ V_{r_i},\ V_{\theta_i},\ m_i \,]^T \tag{4.94}$$

$$\bar{u}_i = [\, u_i,\ \beta_i \,]^T \tag{4.95}$$

The midpoint control used in the Hermite-Simpson defect scheme is now denoted by $\hat{u}_i$ and has the same form as Equation (4.95).

The dynamic equations used for the points located within the Earth-coordinate system have right-hand sides given by

$$f_j = f(t_j, \bar{r}_j, \bar{u}_j) \tag{4.96}$$

$$= \begin{bmatrix} V_{r_j} \\ V_{\theta_j}/r_j \\ \frac{V_{\theta_j}^2}{r_j} - \frac{\mu_e}{r_j^2} + \mu_m \cos\theta_j \left(\frac{d}{r_{m_j}^3} - \frac{1}{d^2}\right) - \mu_m \frac{r_j}{r_{m_j}^3} + \omega^2 r_j + 2\omega V_{\theta_j} \\ \qquad + \frac{c\beta_j}{m_j} \sin u_j \\ -\frac{V_{r_j} V_{\theta_j}}{r_j} - \mu_m \sin\theta_j \left(\frac{d}{r_{m_j}^3} - \frac{1}{d^2}\right) - 2\omega V_{r_j} + \frac{c\beta_j}{m_j} \cos u_j \\ -\beta_j \end{bmatrix} \tag{4.97}$$

where $r_{m_j} = \sqrt{r_j^2 + d^2 - 2 r_j d \cos\theta_j}$ and with a state Jacobian of

$$\frac{\partial f_j}{\partial \bar{r}_j} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ -\frac{V_{\theta_j}}{r_j^2} & 0 & 0 & \frac{1}{r_j} & 0 \\ \frac{\partial \dot{V}_{r_j}}{\partial r_j} & \frac{\partial \dot{V}_{r_j}}{\partial \theta_j} & 0 & \frac{2V_{\theta_j}}{r_j} + 2\omega & -\frac{c\beta_j}{m_j^2} \sin u_j \\ \frac{\partial \dot{V}_{\theta_j}}{\partial r_j} & \frac{\partial \dot{V}_{\theta_j}}{\partial \theta_j} & -\frac{V_{\theta_j}}{r_j} - 2\omega & -\frac{V_{r_j}}{r_j} & -\frac{c\beta_j}{m_j^2} \cos u_j \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.98}$$

where

$$\frac{\partial \dot{V}_{r_j}}{\partial r_j} = -\frac{V_{\theta_j}^2}{r_j^2} + \frac{2\mu_e}{r_j^3} + \frac{\mu_m}{r_{m_j}^5} \left[3(d\cos\theta_j - r_j)^2 - r_{m_j}^2\right] + \omega^2 \tag{4.99}$$

$$\frac{\partial \dot{V}_{r_j}}{\partial \theta_j} = \mu_m \sin\theta_j \left\{ \frac{1}{d^2} - \frac{d}{r_{m_j}^5} \left[r_{m_j}^2 + 3r_j \left(d\cos\theta_j - r_j \right) \right] \right\} \tag{4.100}$$

$$\frac{\partial \dot{V}_{\theta_j}}{\partial r_j} = \frac{V_{r_j} V_{\theta_j}}{r_j^2} - 3\mu_m d \sin\theta_j \frac{d\cos\theta_j - r_j}{r_{m_j}^5} \tag{4.101}$$

$$\frac{\partial \dot{V}_{\theta_j}}{\partial \theta_j} = \mu_m \left\{ \frac{\cos\theta_j}{d^2} - \frac{d}{r_{m_j}^5} \left(r_{m_j}^2 \cos\theta_j - 3r_j d \sin^2\theta_j \right) \right\} \tag{4.102}$$

and a control Jacobian of

$$\frac{\partial f_j}{\partial \bar{u}_j} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{c\beta_j}{m_j}\cos u_j & \frac{c}{m_j}\sin u_j \\ -\frac{c\beta_j}{m_j}\sin u_j & \frac{c}{m_j}\cos u_j \\ 0 & -1 \end{bmatrix} \tag{4.103}$$

The dynamic equations used for the points within the Lunar-coordinate system are

$$f_j^{(m)} = f^{(m)}\left(t_j, \bar{r}_j, \bar{u}_j\right) \tag{4.104}$$

$$= \begin{bmatrix} V_{r_j} \\ V_{\theta_j}/r_j \\ \frac{V_{\theta_j}^2}{r_j} - \frac{\mu_m}{r_j^2} - \mu_e \cos\theta_j \left(\frac{d}{r_{e_j}^3} - \frac{1}{d^2} \right) - \mu_m \frac{r_j}{r_{e_j}^3} + \omega^2 r_j + 2\omega V_{\theta_j} \\ \qquad + \frac{c\beta_j}{m_j}\sin u_j \\ -\frac{V_{r_j} V_{\theta_j}}{r_j} + \mu_e \sin\theta_j \left(\frac{d}{r_{e_j}^3} - \frac{1}{d^2} \right) - 2\omega V_{r_j} + \frac{c\beta_j}{m_j}\cos u_j \\ -\beta_j \end{bmatrix} \tag{4.105}$$

where $r_{e_j} = \sqrt{r_j^2 + d^2 + 2r_j d\cos\theta_j}$ and with a state Jacobian identical to the Earth-centered Jacobian except for the partials which are

$$\frac{\partial \dot{V}_{r_j}}{\partial r_j} = -\frac{V_{\theta_j}^2}{r_j^2} + \frac{2\mu_m}{r_j^3} + \frac{\mu_e}{r_{e_j}^5} \left[3\left(d\cos\theta_j + r_j \right)^2 - r_{e_j}^2 \right] + \omega^2 \tag{4.106}$$

$$\frac{\partial \dot{V}_{r_j}}{\partial \theta_j} = \mu_e \sin\theta_j \left\{ \frac{1}{d^2} - \frac{d}{r_{e_j}^5} \left[3r_j \left(d\cos\theta_j + r_j \right) - r_{e_j}^2 \right] \right\} \tag{4.107}$$

$$\frac{\partial \dot{V}_{\theta_j}}{\partial r_j} = \frac{V_{r_j} V_{\theta_j}}{r_j^2} - 3\mu_m d \sin\theta_j \frac{d\cos\theta_j + r_j}{r_{e_j}^5} \tag{4.108}$$

$$\frac{\partial \dot{V}_{\theta_j}}{\partial \theta_j} = \mu_m \left\{ -\frac{\cos\theta_j}{d^2} + \frac{d}{r_{e_j}^5} \left(r_{e_j}^2 \cos\theta_j + 3 r_j d \sin^2\theta_j \right) \right\} \tag{4.109}$$

and an identical control Jacobian as for the Earth-centered equations.

The general problem statement of a MINLP algorithm splits the real and integer variables into two design vectors where x is the vector of real variables of size n and y is the vector of integer variables of size p. The form of this design vector for the switching function approximation is

$$x = [\, \bar{r}_o,\ u_o,\ \bar{r}_1,\ u_1,\ \ldots\ \bar{r}_f,\ u_f \,]^T \tag{4.110}$$

$$y = \left[\, \bar{\beta}_o,\ \bar{\beta}_1,\ \ldots\ \bar{\beta}_f \,\right]^T \tag{4.111}$$

where $\bar{\beta}_j$ is restricted to be either zero or one and the mass flow at a discrete point is then

$$\beta_j = \beta_{\max}\ \bar{\beta}_j \tag{4.112}$$

The application of the MINLP algorithms result in different subproblems being formed which alter the form of the integer design vector, y, in three basic ways:

y free and bounded This is a relaxed form of the main problem. The integer variables are treated as real variables and allowed to vary between zero and one. This case occurs in both the Branch-and-Bound and GCD algorithms.

y fixed This is the primal subproblem of the GCD algorithm described in Appendix A and results in a pure NLP problem.

y partially-fixed The Branch-And-Bound algorithm of Chapter 2 will fix an arbitrary element of the integer vector to form two branches which will then be investigated. All of the other integer variables will either be free to vary or also be fixed at a value.

The design vectors of the subproblems will then have three main forms. The first case where y is free and bounded will result in a design vector of

$$x_{free} = [\ \bar{r}_o,\ u_o,\ \beta_o,\ \bar{r}_1,\ u_1,\ \beta_1,\ \ldots\ \bar{r}_f,\ u_f,\ \beta_f\ ]^T \tag{4.113}$$

The corresponding constraint gradient for x_{free} is of the form of Equation (2.32) for the Trapezoid defect scheme and Equation (2.41) for the Hermite-Simpson defect scheme where $n_v = 5$ and $n_u = 2$. The second case where y is fixed decreases the size of the subproblem design vector and has the form

$$x_{fixed} = [\ \bar{r}_o,\ u_o,\ \bar{r}_1,\ u_1,\ \ldots\ \bar{r}_f,\ u_f,\ ]^T \tag{4.114}$$

and also has Equation (2.32) and Equation (2.41) for the Trapezoid and Hermite-Simpson defect schemes, respectively. The control size is decreased by one since the mass flow is no longer a control for this subproblem. The third subproblem design vector requires that one or more of the mass flow control elements be fixed. If we let i be the element that is fixed, that portion of the design vector becomes

$$x_{part} = [\ \ldots\ \bar{r}_{i-1},\ u_{i-1},\ \beta_{i-1},\ \bar{r}_i,\ u_i,\ \bar{r}_{i+1},\ u_{i+1},\ \beta_{i+1},\ \ldots\ ]^T \tag{4.115}$$

where it's assumed that the points before and after allow the mass flow to vary. The constraint gradient has the same basic form as the Equations (2.32) and (2.41) but the fixing of the i-th point removes the corresponding column for that previous control point. If we again let i be the element that is fixed, then the two rows of the constraint gradient where β_i is involved become

$$\nabla c_{part} = \begin{bmatrix} \frac{\partial \zeta_i}{\partial \bar{r}_{i-1}} & \frac{\partial \zeta_i}{\partial u_{i-1}} & \frac{\partial \zeta_i}{\partial \beta_{i-1}} & \frac{\partial \zeta_i}{\partial \bar{r}_i} & \frac{\partial \zeta_i}{\partial u_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \zeta_{i+1}}{\partial \bar{r}_i} & \frac{\partial \zeta_{i+1}}{\partial u_i} & \frac{\partial \zeta_{i+1}}{\partial \bar{r}_{i+1}} & \frac{\partial \zeta_{i+1}}{\partial u_{i+1}} & \frac{\partial \zeta_{i+1}}{\partial \beta_{i+1}} \end{bmatrix} \tag{4.116}$$

where the surrounding points are again assumed to vary. The column elimination is repeated whenever another integer variable is fixed.

The computer implementation of the MINLP algorithm must take into account the three possibilities for the design vector. I created an auxiliary bookkeeping vector which was passed into the objective function, constraint evaluation, gradient evaluation, etc. subroutines. This auxiliary vector simply denoted if an integer variable was fixed or free and was used by both the optimization and function routines. The class library placed this vector as part of the integer optimization object with an accessibility of private. Cross-object sharing was done by the friends mechanism or direct inheritance.

Coordinate System Transformation

The design vectors presented above must be modified to handle the two coordinate systems being used. The design vectors include a new state vector which is essentially the same state as the previous point but in the other coordinate system. The trapezoid design vector is

$$x = \left[\bar{r}_o,\ \bar{u}_o,\ \ldots\ ,\ \bar{r}_k,\ \bar{u}_k,\ \bar{r}_k^{(m)},\ \bar{u}_k^{(m)},\ \ldots\ ,\bar{r}_{n_s-1}^{(m)},\ \bar{u}_{n_s-1}^{(m)},\ \bar{r}_f^{(m)},\ \bar{u}_f^{(m)} \right]^T \tag{4.117}$$

and the Hermite-Simpson design vector is

$$x = \left[\ldots\ ,\ \bar{r}_k,\ \bar{u}_k,\ \hat{u}_k,\ \bar{r}_k^{(m)},\ \bar{u}_k^{(m)},\ \hat{u}_k^{(m)},\ \ldots \right]^T \tag{4.118}$$

where the (m) superscript indicate states in the Lunar coordinate system. Note that an extra set of control vectors is needed for the point k since the next defect, i.e., ζ_{k+1}, is a function of the state and control at $k+1$ and at k. The constraint vector is then defined as

$$c(x) = \left[\psi_o,\ \zeta_1,\ \ldots\ ,\zeta_k,\ \tilde{\zeta}_k,\ \zeta_{k+1},\ \ldots\ ,\ \zeta_f,\ \psi_f \right]^T = 0 \tag{4.119}$$

where

$$\tilde{\zeta}_k = \bar{r}_k^{(m)} - T\left(\bar{r}_k\right) \tag{4.120}$$

where $T\left(\bar{r}\right)$ is the transformation of Equations (3.90) - (3.93). The constraint gradients are essentially the same as presented in Chapter 2 for the majority of each matrix. The

alteration takes place at the k-th point where the new defect, $\tilde{\zeta}$, is introduced. This extra defect adds a new row to the constraint gradient and for the trapezoid scheme is

$$\nabla c = \begin{bmatrix} \ddots & \ddots & \ddots & \ddots & & & & & \\ & \frac{\partial \zeta_k}{\partial \bar{r}_{k-1}} & \frac{\partial \zeta_k}{\partial \bar{u}_{k-1}} & \frac{\partial \zeta_k}{\partial \bar{r}_k} & \frac{\partial \zeta_k}{\partial \bar{u}_k} & & & & \\ & & & \frac{\partial \tilde{\zeta}_k}{\partial \bar{r}_k} & 0 & \frac{\partial \tilde{\zeta}_k}{\partial \bar{r}_k^{(m)}} & & & \\ & & & & & \frac{\partial \zeta_{k+1}}{\partial \bar{r}_k^{(m)}} & \frac{\partial \zeta_{k+1}}{\partial \bar{u}_k^{(m)}} & \frac{\partial \zeta_{k+1}}{\partial \bar{r}_{k+1}^{(m)}} & \frac{\partial \zeta_{k+1}}{\partial \bar{u}_{k+1}^{(m)}} \\ & & & & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.121}$$

and for the Hermite-Simpson defect scheme is

$$\nabla c = \begin{bmatrix} \ddots & \ddots & \ddots & & & \\ \frac{\partial \zeta_k}{\partial \bar{r}_k} & \frac{\partial \zeta_k}{\partial \bar{u}_k} & \frac{\partial \zeta_k}{\partial \hat{u}_k} & & & \\ \frac{\partial \tilde{\zeta}_k}{\partial \bar{r}_k} & 0 & 0 & \frac{\partial \tilde{\zeta}_k}{\partial \bar{r}_k^{(m)}} & & \\ & & & \frac{\partial \zeta_{k+1}}{\partial \bar{r}_k^{(m)}} & \frac{\partial \zeta_{k+1}}{\partial \bar{u}_k^{(m)}} & \frac{\partial \zeta_{k+1}}{\partial \hat{u}_k^{(m)}} \\ & & & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.122}$$

The partial of the new linking defect is

$$\frac{\partial \tilde{\zeta}}{\partial \bar{r}_k} = -\frac{\partial P(\bar{r}_k)}{\partial \bar{r}_k} \tag{4.123}$$

$$\frac{\partial \tilde{\zeta}}{\partial \bar{r}_k^{(m)}} = \mathrm{I} \tag{4.124}$$

and if $\hat{\bar{r}} = P(\bar{r})$ then

$$\frac{\partial \hat{r}_k}{\partial r_k} = \frac{r_k - d\cos\theta_k}{\hat{r}_k} \tag{4.125}$$

$$\frac{\partial \hat{r}_k}{\partial \theta_k} = \frac{r_k d \sin\theta_k}{\hat{r}_k} \tag{4.126}$$

$$\frac{\partial \hat{\theta}_k}{\partial r_k} = \left[1 - \left(\frac{r_k \sin\theta_k}{\hat{r}_k}\right)^2\right]^{-1/2} \frac{\sin\theta_k}{\hat{r}_k} \tag{4.127}$$

$$\frac{\partial \hat{\theta}_k}{\partial \theta_k} = \left[1 - \left(\frac{r_k \sin\theta_k}{\hat{r}_k}\right)^2\right]^{-1/2} \frac{r_k \cos\theta_k}{\hat{r}_k} \tag{4.128}$$

$$\frac{\partial \hat{V}_{r_k}}{\partial r_k} = \frac{\hat{r}_k V_{r_k} - \hat{V}_{r_k}\left(r_k - d\cos\theta_k\right)}{\hat{r}_k^2} \tag{4.129}$$

$$\frac{\partial \hat{V}_{r_k}}{\partial \theta_k} = \frac{d\sin\theta_k\left(V_{r_k}\hat{r}_k - \hat{V}_{r_k} r_k\right) + \hat{r}_k d\cos\theta_k V_{\theta_k}}{\hat{r}_k^2} \tag{4.130}$$

$$\frac{\partial \hat{V}_{r_k}}{\partial V_{r_k}} = \frac{r_k - d\cos\theta_k}{\hat{r}_k} \tag{4.131}$$

$$\frac{\partial \hat{V}_{r_k}}{\partial V_{\theta_k}} = \frac{d\sin\theta_k}{\hat{r}_k} \tag{4.132}$$

$$\frac{\partial \hat{V}_{\theta_k}}{\partial r_k} = \frac{\hat{r}_k V_{\theta_k} - \hat{V}_{\theta_k}\left(r_k - d\cos\theta_k\right)}{\hat{r}_k} \tag{4.133}$$

$$\frac{\partial \hat{V}_{\theta_k}}{\partial \theta_k} = \frac{d\sin\theta_k\left(\hat{r}_k V_{\theta_k} - r_k \hat{V}_{\theta_k}\right) - \hat{r}_k d\cos\theta_k V_{r_k}}{\hat{r}_k^2} \tag{4.134}$$

$$\frac{\partial \hat{V}_{\theta_k}}{\partial V_{r_k}} = -\frac{d\sin\theta_k}{\hat{r}_k} \tag{4.135}$$

$$\frac{\partial \hat{V}_{\theta_k}}{\partial V_{\theta_k}} = \frac{r_k - d\cos\theta_k}{\hat{r}_k} \tag{4.136}$$

The coordinate transformation for the MINLP approximation can then be applied by using the design vectors (4.117) and (4.118), the constraint vector (4.119), the transformation definition (4.120), and the constraint gradients (4.121) and (4.122). This form of the coordinate transformation turns out to be very stable and provides excellent convergence characteristics mainly due to the analytic equations for the partials of the transformation. Numerical experiments which relied on finite-difference approximations always introduced error and sometimes prevented fast convergence near the solution.

Fanning

The ability of a MINLP algorithm to successfully converge to a solution is highly problem dependent. Several types of problems in process synthesis and chemical reac-

tions can solve MINLP problems of several hundred integer and thousands of floating variables, while computer networking and scheduling systems can rarely be solved for an integer system of greater than twenty. It is anticipated that the discretization methods will require on the order of hundreds of discrete points in order to satisfy the equations of motion to a reasonable degree and that it will be difficult for a MINLP problem to solve the switching approximation for such a large number. To reduce the number of integer variables and increase the chances of a successful optimal search, a numerical method called "fanning" is now introduced. This is a new technique developed explicity for a collocation solution to a trajectory problem. The fanning method uses an integer variable over a range of discretized points. The number of discrete trajectory points assigned to a particular integer is called its fan and is allowed to vary for each integer variable. The fan ratio is defined as

$$\text{fan ratio} \quad = \frac{\text{number of discrete trajectory points}}{\text{number of integer variables}} \tag{4.137}$$

The fan ratio is then bounded below by one and above by the number of trajectory points with a fan ratio of one indicating the same number of integer variables as discrete points.

An auxiliary fan vector, Φ, is defined as

$$\Phi = [\ \phi_1,\ \phi_2,\ \ldots\ \phi_p\ ]^T \tag{4.138}$$

where each component represents the fan of the corresponding integer variable so Φ is the same length as y. The integer variable β_i will fan into ϕ_i discrete points where the first point will have an index of

$$\sum_{j=0}^{i} \phi_j \tag{4.139}$$

and the final index will be

$$\sum_{j=0}^{i+1} \phi_j - 1 \tag{4.140}$$

A quick example will illustrate the idea. Consider a three integer parameter vector y with an auxiliary fan vector of

$$\Phi = [\ 2,\ 5,\ 2\ ]^T \tag{4.141}$$

The number of discrete points is then 9 or the sum of all the fans. The massflow controls for the discrete points are then

$$\beta_0 = \beta_{\max}\bar{\beta}_1 \tag{4.142}$$

$$\beta_1 = \beta_{\max}\bar{\beta}_1 \tag{4.143}$$

$$\beta_2 = \beta_{\max}\bar{\beta}_2 \tag{4.144}$$

$$\beta_3 = \beta_{\max}\bar{\beta}_2 \tag{4.145}$$

$$\beta_4 = \beta_{\max}\bar{\beta}_2 \tag{4.146}$$

$$\beta_5 = \beta_{\max}\bar{\beta}_2 \tag{4.147}$$

$$\beta_6 = \beta_{\max}\bar{\beta}_2 \tag{4.148}$$

$$\beta_7 = \beta_{\max}\bar{\beta}_3 \tag{4.149}$$

$$\beta_8 = \beta_{\max}\bar{\beta}_3 \tag{4.150}$$

The initial index where the second integer variable is used is $\beta_2 = \beta_{\phi_1}$ and the final index is $\beta_6 = \beta_{\phi_1+\phi_2-1}$. The distribution of the fans are easily implemented and do not present a large amount of bookkeeping other than the auxiliary fan vector. The distribution of the fans between a given number of integer variables is accomplished by the solution of the relaxed subproblem of x_{free}. The solution of this subproblem provides a mass flow control history to estimate where and when switching is likely to occur. The initial portions of the departure trajectory will not likely allow a switch to occur and the relaxed solution for the mass flow reflects this by staying at or near the maximum value. The translunar coast portion of the transfer also prevents switching and the mass flow is at or near zero. The fans for these portions of the trajectories can therefore be quite large without affecting the accuracy of the MINLP switching approximation. The

general rule I used for this work was to place the first fan over the initial portion of the trajectory where the mass flow did not decrease below 90% of its maximum value. A corresponding large fan is placed at the translunar coast where the mass flow does not increase over 10% of its minimum value. The number of integer variables which will accurately approximate the switching function can be estimated by this procedure. The relaxed subproblem mass flow control history is analyzed and whenever the mass flow changes by more than 2% of its current value a new integer variable is introduced with a corresponding fan. This approach proved to work very well for the Lunar transfers studied in this chapter. A numerical example at the end of this section will help to clarify many of these points.

The effect of both the different subproblem design vectors and coordinate transformations on the use of fanning is straightforward. The design vectors where part of the integer variables are fixed simply eliminate more then one row of the design vector and more than one column of the constraint gradient. The coordinate transformation, on the other hand, requires that the fan before the coordinate switch and after terminate on that point. This prevents convergence problems when the minimizing algorithm cannot change the linking control histories separately.

Fanning has two possible detrimental effects on the solution of a MINLP trajectory. The first and more severe is the prevention of the MINLP algorithm from converging to a solution when the fan ratio is very large. This nearly always occurs when using the Branch-and-Bound method described in Chapter 2. No subproblems of any branch return a feasible solution which results in the method terminating with an error. This is a reasonable expectation since a large fan ratio forces the trajectory into odd thrusting and coast periods. The second effect also results from a large fan ratio and has to do with the quality of the switching approximation when a MINLP solution is found. The goal of using MINLP is to both gain an estimate of the performance of a switching transfer and an accurate idea of the number of switches occurring during a switching

transfer for a prescribed time-of-flight. A large fan ratio prevents both of these goals.

Mesh Allocation Strategies

The general discretization methods allow the points to be placed anywhere along the time axis. An automatic placement is discussed briefly in Chapter 2 and specific implementations are presented here. The goal of automatic allocation is to accurately model the MINLP switching approximation. Three different allocation functions were considered:

1. The first placement strategy attempts to place more points where the polar angle is changing rapidly and assumes the form,

$$\gamma_\theta = \frac{1}{\Delta} + \left| \frac{d\theta/dt}{\theta\delta} \right| \tag{4.151}$$

where the points will be placed near the Earth and Moon because of the spirals of both the departure and capture.

2. The second placement strategy monitors the current total energy level and places points near where the energy is changing rapidly,

$$\gamma_E = \frac{1}{\Delta} + \left| \frac{dE/dt}{E\delta} \right| \tag{4.152}$$

where

$$E = \frac{V_r^2 + V_\theta^2}{2} - \frac{\mu}{r} \tag{4.153}$$

thus

$$\frac{dE}{dt} = V_r \dot{V}_r + V_\theta \dot{V}_\theta + \frac{\mu}{r^2}\dot{r} \tag{4.154}$$

Since the thrusting portions of the trajectory are causing the energy change, this strategy also places points where switching is likely to occur. It also has the advantage of not placing points towards the beginning of the Earth departure since the spacecraft has not attained enough energy to cause switching.

3. The third allocation method attempts to place points where the total velocity of the trajectory is changing rapidly,

$$\gamma_V = \frac{1}{\Delta} + \left| \frac{dV/dt}{V\delta} \right| \tag{4.155}$$

where

$$V = \sqrt{V_r^2 + V_\theta^2} \tag{4.156}$$

thus

$$\frac{dV}{dt} = V_r \dot{V}_r + V_\theta \dot{V}_\theta \tag{4.157}$$

The logic for this strategy is essentially the same as γ_E but with the idea that the velocity is a more accurate measurement of when switching will occur rather than energy.

Numerical Example

A numerical example shows the aspects discussed in this section. The solution for the switching transfer of 8.5 days is solved using the MINLP approximation. Both defect schemes were applied in the manner suggested earlier in this section and Table 4.1 shows the numerical parameters for the problem. The CPU times given are for a SGI Indy workstation under load sharing. The number of discrete points for both defect schemes was initially set at 750 and decreased using the selection process described in Chapter 2 by Betts and Huffman [16]. The number of points for the Trapezoid scheme was reduced to 605 and the number of points for the Hermite-Simpson scheme was reduced to 562. The mass flow control solution for the relaxed subproblem is shown in Figure 4.7 for the Earth-centered coordinate system. The relaxed problem converged to a solution in 56 iterations. The number of thrust and coast arcs is easily seen from this relaxed solution, but the exact switching times or even the lengths of the separate phases is difficult to distinguish. However, the relaxed problem is generally solvable and can be used for the

Table 4.1 MINLP Numerical Example for a Flight Time of 8.5 Days

		Defect Scheme	
		Trapezoid	Hermite-Simpson
γ_θ	function evaluations	890	1,020
	gradient evaluations	130	176
	CPU time, sec	934	1,133
	m_f/m_i	0.9350	0.9390
γ_E	function evaluations	765	842
	gradient evaluations	115	133
	CPU time, sec	855	989
	m_f/m_i	0.9401	0.9405
γ_V	function evaluations	798	1,122
	gradient evaluations	105	201
	CPU time, sec	888	1,012
	m_f/m_i	0.9400	0.9425

MINLP algorithms already described. The number of integer variables was fixed at fifty and then distributed over the trajectory as suggested earlier. The fan ratios for the Trapezoid and Hermite-Simpson defect schemes are 12.1 and 11.24, respectively.

Figures 4.8 and 4.9 are an attempt to show the benefits of a collocation type of method over a terminal error function method. Figure 4.8 is a portion of the initial estimate for the problem obtained from the scaled thrust-coast-thrust solution from Chapter 2. One hundred points are shown which takes place during the ninth and tenth spirals of the initial estimate. The points are evenly distributed over time as can be seen by the "bunching" of points near the end of the trajectory portion. Figure 4.9 shows the same number and location of points for both defect scheme solutions as well as the exact solution. Two improvements can be seen from these trajectories. The first is the inclusion of the coast arc which is difficult in a non-collocation method and the second is the point placement. Note the dense distribution of points, for both schemes, about the points of engine switching. The energy distribution scheme, γ_E, is used for the results shown in Figure 4.9 and when compared to the even distribution of the initial estimate in Figure 4.8, γ_E can be seen to provide an excellent point distribution for a switching

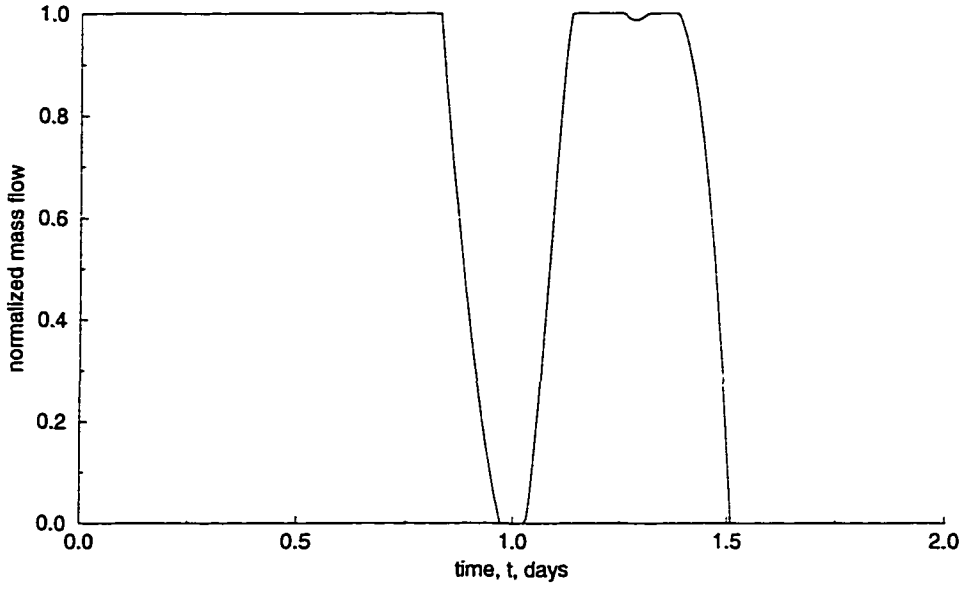

Figure 4.7 Relaxed Mass flow Control History for the Example MINLP Solution

trajectory where the Hermite-Simpson defect scheme provides a switching time closer to the "true" solution.

Solution of the 2PBVP from MINLP Approximation

The solution of the switching 2PBVP from the MINLP approximation is done by a relaxation technique which requires an initial approximation of the combined state-costate system. Excellent estimates for the states are provided by the MINLP solution, and an estimate for the costates will be derived which utilizes the control points returned from the MINLP solution. The time-derivative of the mass costate was found to be

$$\dot{\lambda}_m = -\frac{c\beta}{m^2}|\lambda_V(t)| \tag{4.158}$$

Since $c|\lambda_V|/m^2 > 0$ and β is either 0 or $\beta_{\max}$, the mass costate derivative is always zero or some negative value and λ_m always decreases ($\beta = \beta_{\max}$) or remains constant ($\beta = 0$).

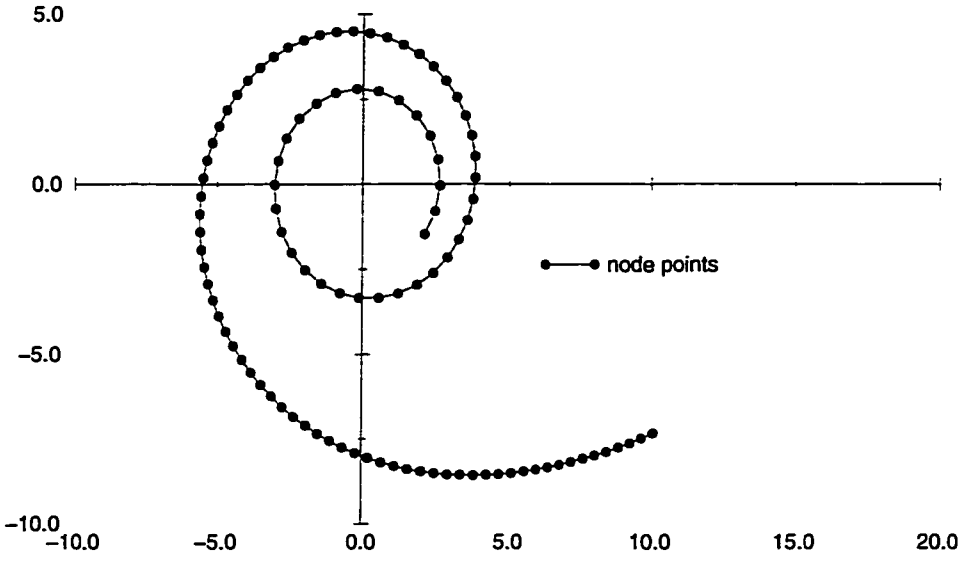

Figure 4.8 Initial Estimate for the Example MINLP Problem

A piece-wise linear approximation to the mass costate can then provide a good initial estimate for the mass costate time history. Figure 4.10 shows this approximation for a coast, thrust, and coast phase of the trajectory. The sloping portion of the figure is the thrust phase where $j-1$ denotes the beginning of the thrust phase and j denotes the end. A linear slope between the two points can be written as:

$$M_j = \frac{\lambda_m(t_j) - \lambda_m(t_{j-1})}{t_j - t_{j-1}} \tag{4.159}$$

The only values of the switching function which are known are at the switching times where the function must be zero. Setting Equation (4.36) or (4.38) to zero provides the mass costate at that switching time in terms of the norm of the costate velocities,

$$\lambda_m(t_j) = -\frac{c}{m(t_j)} |\lambda_V(t_j)| \tag{4.160}$$

The linear slope approximation then becomes:

$$M_j = \frac{-c}{t_j - t_{j-1}} \left[\frac{|\lambda_V(t_j)|}{m(t_j)} - \frac{|\lambda_V(t_{j-1})|}{m(t_{j-1})} \right] \tag{4.161}$$

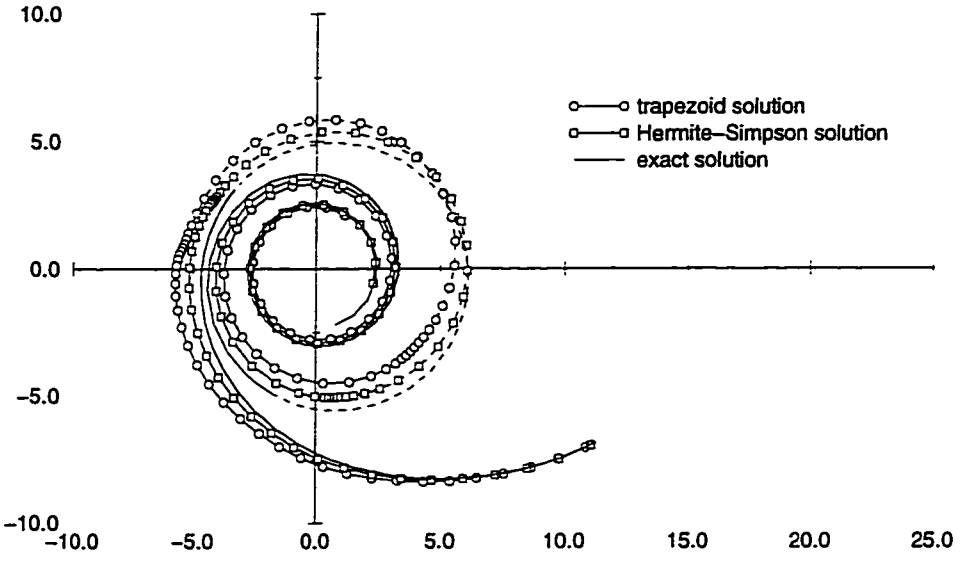

Figure 4.9 MINLP Trajectory Solutions for the Example MINLP Problem

Equations (4.158) and (4.161) can be used to solve for the costate norm

$$|\lambda_V(t_{j-1})| = \frac{m^2(t_{j-1})}{m(t_j)\left[\beta(t_j - t_{j-1}) + m(t_{j-1})\right]} |\lambda_V(t_j)| \tag{4.162}$$

An approximation for the costate velocity norm at the beginning of the thrust phase can then be determined if an approximation for the costate velocity norm is known at the end of that phase. A starting point is needed for this process and this is provided by the final boundary condition on the mass costate. The slope for the final stage is given by

$$M_f = \frac{\lambda_m(t_f) - \lambda_m(t_{f-1})}{t_f - t_{f-1}} \tag{4.163}$$

$$= \frac{-1 - c|\lambda_V(t_{f-1})|/m(t_{f-1})}{t_f - t_{f-1}} \tag{4.164}$$

where t_{f-1} is the last switching time before the end of the trajectory. In Equation (4.164), Equation (4.69) is used to set $\lambda_m(t_f)$ to -1 and Equation (4.160) is used to replace $\lambda_m(t_{f-1})$. Combining M_f with Equation (4.158) and solving for the costate velocity

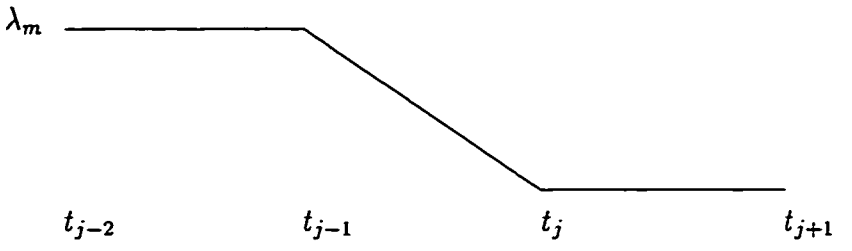

Figure 4.10 Mass Costate Approximation for Interior Point

norm, we obtain

$$|\lambda_V(t_{f-1})| = \frac{m^2(t_{f-1})}{c[\beta(t_f - t_{f-1}) + m(t_{f-1})]} \tag{4.165}$$

The values of the costate velocity norms can then be approximated at each switching time and a curve-fit of these points can provide an estimate of the complete time history. The boundary points for $|\lambda_V|$ can be found by extrapolating from the interior point estimates. The mass costate time history is then approximated by Equation (4.160) where the costate velocity norms are now known. The individual velocity costates are then easily determined from the thrust steering angle control provided by the MINLP solution, i.e.,

$$\lambda_{V_r} = -|\lambda_V| \sin u \tag{4.166}$$

$$\lambda_{V_\theta} = -|\lambda_V| \cos u \tag{4.167}$$

where the optimal trigonometric relations for the velocity costates are known from Equations 4.32 - 4.35.

The final two costate time histories, $\lambda_r(t)$ and $\lambda_\theta(t)$, are found using Equation (4.79). Since the switching function of Equation (4.36) or (4.38) is determined by the mass costate and costate velocity norm, a numerical approximation of the switching function derivative can be found from the estimates detailed in the preceding paragraphs. It is not possible to completely determine both the radial costate and polar angle costate

since only one equation relates the two. However, the polar angle costate is known to start at zero and terminate at zero. Also, experience shows that λ_θ does not grow very large and usually has an order of magnitude of 10^{-2} or less. The polar angle costate is then set to a very small constant value, approximately 10^{-3}, for the entire trajectory. The radial costate can then be found using the numerical value of the switching function derivative and Equation (4.79).

The relaxation solution method requires Jacobian information of the system to be solved. This Jacobian information is used to change the discretized points by the method presented in Chapter 2 and given by Press, et. al. [22]. The combined state-costate system can be written as

$$\bar{R} = \begin{bmatrix} \bar{r} \\ \bar{\lambda} \end{bmatrix} \tag{4.168}$$

with their respective dynamics as

$$\dot{\bar{R}} = F = \begin{bmatrix} f \\ -\frac{\partial f}{\partial \bar{r}}^T \lambda \end{bmatrix} \tag{4.169}$$

where f is the vector state rate for the restricted circular three-body problem in either the Earth or Lunar-centered coordinate system. The combined Jacobian can be expressed by

$$\frac{\partial F}{\partial \bar{R}} = \begin{bmatrix} \frac{\partial f}{\partial \bar{r}} & \frac{\partial f}{\partial \lambda} \\ \frac{\partial \dot{\lambda}}{\partial \bar{r}} & -\frac{\partial f}{\partial \bar{r}}^T \end{bmatrix} \tag{4.170}$$

The upper left and lower right portions of this matrix have already been determined for both the Earth and Lunar centered three-body equations. The upper right portion is the Jacobian of the state equations with respect to the costate variables which were used to replace the control. The control form is identical for both coordinate systems and can be written as

$$\sin u = \frac{-\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{4.171}$$

$$\cos u = \frac{-\lambda_{V_\theta}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{4.172}$$

$$\tag{4.173}$$

where the "m" subscript would be used for the Lunar variables. There are then only four non-zero elements for that portion and these are found to be

$$\frac{\partial \dot{V}_r}{\partial \lambda_{V_r}} = -\frac{c\beta}{m} \frac{\lambda_{V_\theta}^2}{\left(\lambda_{V_r}^2 + \lambda_{V_\theta}^2\right)^{3/2}} \tag{4.174}$$

$$\frac{\partial \dot{V}_r}{\partial \lambda_{V_\theta}} = \frac{c\beta}{m} \frac{\lambda_{V_r}\lambda_{V_\theta}}{\left(\lambda_{V_r}^2 + \lambda_{V_\theta}^2\right)^{3/2}} \tag{4.175}$$

$$\frac{\partial \dot{V}_\theta}{\partial \lambda_{V_r}} = \frac{c\beta}{m} \frac{\lambda_{V_r}\lambda_{V_\theta}}{\left(\lambda_{V_r}^2 + \lambda_{V_\theta}^2\right)^{3/2}} \tag{4.176}$$

$$\frac{\partial \dot{V}_\theta}{\partial \lambda_{V_\theta}} = -\frac{c\beta}{m} \frac{\lambda_{V_r}^2}{\left(\lambda_{V_r}^2 + \lambda_{V_\theta}^2\right)^{3/2}} \tag{4.177}$$

$$\tag{4.178}$$

The lower left portion of Equation (4.170) consists of partial derivatives of the costate equations.

$$\frac{\partial \dot{\lambda}_r}{\partial r} = \frac{-2V_\theta}{r^3}\lambda_\theta - \frac{\partial^2 \dot{V}_r}{\partial r^2}\lambda_{V_r} - \frac{\partial^2 \dot{V}_\theta}{\partial r^2}\lambda_{V_\theta} \tag{4.179}$$

$$\frac{\partial \dot{\lambda}_\theta}{\partial r} = -\frac{\partial^2 \dot{V}_r}{\partial r \partial \theta}\lambda_{V_r} - \frac{\partial^2 \dot{V}_\theta}{\partial r \partial \theta}\lambda_{V_\theta} \tag{4.180}$$

$$\frac{\partial \dot{\lambda}_{V_r}}{\partial r} = -\frac{V_\theta}{r^2}\lambda_{V_\theta} \tag{4.181}$$

$$\frac{\partial \dot{\lambda}_{V_\theta}}{\partial r} = \frac{\lambda_\theta}{r^2} + \frac{2V_\theta}{r^2}\lambda_{V_r} - \frac{V_r}{r^2}\lambda_{V_\theta} \tag{4.182}$$

$$\frac{\partial \dot{\lambda}_m}{\partial r} = 0 \tag{4.183}$$

$$\frac{\partial \dot{\lambda}_r}{\partial \theta} = -\frac{\partial^2 \dot{V}_r}{\partial \theta \partial r}\lambda_{V_r} - \frac{\partial^2 \dot{V}_\theta}{\partial \theta \partial r}\lambda_{V_\theta} \tag{4.184}$$

$$\frac{\partial \dot{\lambda}_\theta}{\partial \theta} = -\frac{\partial^2 \dot{V}_r}{\partial \theta^2}\lambda_{V_r} - \frac{\partial^2 \dot{V}_\theta}{\partial \theta^2}\lambda_{V_\theta} \tag{4.185}$$

$$\frac{\partial \dot{\lambda}_{V_r}}{\partial \theta} = 0 \tag{4.186}$$

$$\frac{\partial \dot{\lambda}_{V_\theta}}{\partial \theta} = 0 \tag{4.187}$$

$$\frac{\partial \dot{\lambda}_m}{\partial \theta} = 0 \tag{4.188}$$

$$\frac{\partial \dot{\lambda}_r}{\partial V_r} = -\frac{V_\theta \lambda_{V_\theta}}{r^2} \tag{4.189}$$

$$\frac{\partial \dot{\lambda}_\theta}{\partial V_r} = 0 \tag{4.190}$$

$$\frac{\partial \dot{\lambda}_{V_r}}{\partial V_r} = 0 \tag{4.191}$$

$$\frac{\partial \dot{\lambda}_{V_\theta}}{\partial V_r} = \frac{\lambda_{V_\theta}}{r} \tag{4.192}$$

$$\frac{\partial \dot{\lambda}_m}{\partial V_r} = -\frac{c\beta}{m^2} \frac{\lambda_{V_r}}{\sqrt{\lambda_{V_r}^2 + \lambda_{V_\theta}^2}} \tag{4.193}$$

$$\frac{\partial \dot{\lambda}_r}{\partial V_\theta} = \frac{\lambda_\theta}{r^2} + \frac{2V_\theta \lambda_{V_r}}{r^2} - \frac{V_r \lambda_{V_\theta}}{r^2} \tag{4.194}$$

$$\frac{\partial \dot{\lambda}_\theta}{\partial V_\theta} = 0 \tag{4.195}$$

$$\frac{\partial \dot{\lambda}_{V_r}}{\partial V_\theta} = \frac{\lambda_{V_\theta}}{r} \tag{4.196}$$

$$\frac{\partial \dot{\lambda}_{V_\theta}}{\partial V_\theta} = \frac{-2\lambda_{V_r}}{r} \tag{4.197}$$

$$\frac{\partial \dot{\lambda}_m}{\partial V_\theta} = -\frac{c\beta}{m^2} \frac{\lambda_{V_r}}{\sqrt{\lambda_{V_\theta}^2 + \lambda_{V_\theta}^2}} \tag{4.198}$$

$$\frac{\partial \dot{\lambda}_r}{\partial m} = 0 \tag{4.199}$$

$$\frac{\partial \dot{\lambda}_\theta}{\partial m} = 0 \tag{4.200}$$

$$\frac{\partial \dot{\lambda}_{V_r}}{\partial m} = 0 \tag{4.201}$$

$$\frac{\partial \dot{\lambda}_{V_\theta}}{\partial m} = 0 \tag{4.202}$$

$$\frac{\partial \dot{\lambda}_m}{\partial m} = \frac{2c\beta}{m^3} \sqrt{\lambda_{V_\theta}^2 + \lambda_{V_\theta}^2} \tag{4.203}$$

where

$$\frac{\partial^2 \dot{V}_r}{\partial r^2} = \frac{2V_\theta^2}{r^3} - \frac{6\mu_e}{r^4} - \frac{3\mu_m}{r_m^7} \left[3r_m^2 - 5\left(d\cos\theta - r\right)^2\right] \left(d\cos\theta - r\right) \tag{4.204}$$

$$\frac{\partial^2 \dot{V}_\theta}{\partial r^2} = -\frac{2V_r V_\theta}{r^3} + \frac{3\mu_m d\sin\theta}{r_m^7} \left[r_m^2 - 5\left(d\cos\theta - r\right)^2\right] \tag{4.205}$$

$$\frac{\partial^2 \dot{V}_r}{\partial r \partial \theta} = -3\mu_m \frac{d\sin\theta}{r_m^7}\left[r_m^2\left(2d\cos\theta - 3r\right) + 5r\left(d\cos\theta - r\right)^2\right] \quad (4.206)$$

$$\frac{\partial^2 \dot{V}_\theta}{\partial r \partial \theta} = 3\mu_m \frac{d}{r_m^7}\left\{5rd\sin^2\theta\left(d\cos\theta - r\right) - r_m^2\left[d\left(1 - 2\sin^2\theta\right] - r\cos\theta\right)\right\} \quad (4.207)$$

$$\frac{\partial^2 \dot{V}_r}{\partial \theta \partial r} = \frac{\partial^2 \dot{V}_r}{\partial r \partial \theta} \quad (4.208)$$

$$\frac{\partial^2 \dot{V}_\theta}{\partial \theta \partial r} = \frac{\partial^2 \dot{V}_\theta}{\partial r \partial \theta} \quad (4.209)$$

$$\frac{\partial^2 \dot{V}_r}{\partial \theta^2} = \mu_m\left\{\frac{\cos\theta}{d^2} - \frac{d}{r_m^7}\left[r_m^2\left(\cos\theta\left(d^2 - 2r^2\right) + rd\left(\cos\theta^2 - 6\sin^2\theta\right)\right)\right. \quad (4.210)$$

$$\left. -15r^2 d\sin^2\theta\left(d\cos\theta - r\right)\right]\Big\} \quad (4.211)$$

$$\frac{\partial^2 \dot{V}_\theta}{\partial \theta^2} = -\mu_m\left\{\frac{\sin\theta}{d^2} - \frac{d}{r_m^7}\left[r_m^2\left(r^2 + d^2 + 7rd\cos\theta\right) - 15r^2 d^2\sin^2\theta\right]\right\} \quad (4.212)$$

The Lunar-centered equations produce a different costate system because the partial differential equations of the radial and tangential velocities with respect to the radius and polar angle are different between the two coordinate systems. Equations (4.204) - (4.212) are replaced by their Lunar equivalents:

$$\frac{\partial^2 \dot{V}_{r_m}}{\partial r_m^2} = \frac{2V_{\theta_m}^2}{r_m^3} - \frac{6\mu_m}{r_m^4} + \frac{3\mu_e}{r^7}\left[3r^2 - 5\left(d\cos\theta_m + r_m\right)^2\right]\left(d\cos\theta_m - r_m\right) \quad (4.213)$$

$$\frac{\partial^2 \dot{V}_{\theta_m}}{\partial r_m^2} = -\frac{2V_{r_m}V_{\theta_m}}{r_m^3} - \frac{3\mu_e d\sin\theta_m}{r^7}\left[r^2 - 5\left(d\cos\theta_m + r_m\right)^2\right] \quad (4.214)$$

$$\frac{\partial^2 \dot{V}_{r_m}}{\partial r_m \partial \theta_m} = -3\mu_e \frac{d\sin\theta_m}{r^7}\left[r_m^2\left(2d\cos\theta_m + 3r_m\right) - 5r_m\left(d\cos\theta_m + r_m\right)^2\right] \quad (4.215)$$

$$\frac{\partial^2 \dot{V}_{\theta_m}}{\partial r_m \partial \theta_m} = -3\mu_e \frac{d}{r^7}\left\{5r_m d\sin^2\theta_m\left(d\cos\theta_m + r_m\right)\right. \quad (4.216)$$

$$\left. +r^2\left[d\left(1 - 2\sin^2\theta_m\right) + r\cos\theta_m\right]\right\} \quad (4.217)$$

$$\frac{\partial^2 \dot{V}_{r_m}}{\partial \theta_m \partial r_m} = \frac{\partial^2 \dot{V}_{r_m}}{\partial r_m \partial \theta_m} \quad (4.218)$$

$$\frac{\partial^2 \dot{V}_{\theta_m}}{\partial \theta_m \partial r_m} = \frac{\partial^2 \dot{V}_{\theta_m}}{\partial r_m \partial \theta_m} \quad (4.219)$$

$$\frac{\partial^2 \dot{V}_{r_m}}{\partial \theta^2} = \mu_m\left\{\frac{\cos\theta}{d^2} - \frac{d}{r_m^7}\left[r_m^2\left(\cos\theta\left(d^2 - 2r^2\right) + rd\left(\cos\theta^2 - 6\sin^2\theta\right)\right)\right. \quad (4.220)$$

$$\left. -15r^2 d\sin^2\theta\left(d\cos\theta - r\right)\right]\Big\} \quad (4.221)$$

$$\frac{\partial^2 \dot{V}_{\theta_m}}{\partial \theta^2} = -\mu_m\left\{\frac{\sin\theta}{d^2} - \frac{d}{r_m^7}\left[r_m^2\left(r^2 + d^2 + 7rd\cos\theta\right) - 15r^2 d^2\sin^2\theta\right]\right\} \quad (4.222)$$

General Solution Strategy

A general methodology of solving for an optimal switching transfer using the techniques outlined in this chapter is presented in this section. A specific mission profile and a fixed time-of-flight are all that are required for an optimal switching transfer solution. An outline of the solution strategy follows:

1. Initial estimate calculation
2. MINLP approximate solution found
   (a) Estimate preprocessing
   (b) Trapezoid solution
   (c) Hermite-Simpson solution
   (d) Solution post-processing
3. 2PBVP solution by the relaxation method
4. 2PBVP solution by terminal error method

Step 1 involves the techniques of Chapter 3 for determining a thrust-coast-thrust trajectory. The mission profile is set in this step, i.e., the initial mass and thrust-to-weight ratio, the parking orbits, etc. The total time-of-flight for this estimate is determined by the method and the coordinate transformation cross-over point is passed on to the next step. Step 2 uses the MINLP strategy of this chapter to determine an estimate of the switching Lunar transfer and is further divided into four phases. The first phase manipulates the estimate from Step 1 to put the information into the form that the MINLP setup requires. The state and controls time histories, which are for the initial estimate's time-of-flight, are scaled to fit the desired time-of-flight. The scaling of the "base" thrust-coast-thrust solution time histories for solving of any general transfer time

was found to work most effectively. Numerical experiments using other switching solutions of shorter or longer times-of-flight to provide this base estimate showed a strong tendency to be unstable and had difficulty converging. A probable explanation is the change in the control histories between a switching and non-switching transfer. The non-switching transfer basically thrusts along the flight path angle for most of the departure causing a steady increase in the radial distance from the Earth. A switching transfer also thrusts along the flight path angle but only near the perigee points. The timing of the engine switches must coincide exactly with the time the spacecraft is near the perigee points or severe discrepancies are introduced. This problem is not found in the "base" solution which always attempts to increase the radial distance. Other preprocessing in Phase 2a includes scaling of variables for different distance and time units, a conservative calculation of the number of needed trajectory points, and a distribution of the fan based upon the solution of the relaxed subproblem. Phase 2b solves the MINLP approximation using the Trapezoid discretization scheme. The Trapezoid scheme is more robust and allows convergence of the MINLP algorithm given the estimate from Step 1. Phase 2c solves the MINLP approximation using the Hermite-Simpson discretization scheme where the initial estimate is the solution found by the Trapezoid scheme in Phase 2b. The midpoint values are interpolated from the Trapezoid solutions. The Hermite-Simpson solution provides an excellent estimate of the state and control histories, including the switching times. The last phase, Phase 2d, covers any necessary cleanup of the MINLP solutions such as rescaling of dimensional units and changing the time when the coordinate transformation occurs. Step 3 uses the relaxation method to solve the 2PBVP which is detailed in the last section. This solution provides the costate time histories for the entire trajectory including the initial conditions. The final step, Step 4, solves the 2PBVP detailed in the first section of this chapter using the terminal error function method in much the same way as Chapter 3.

The combined state/costate system consists of twenty dynamic equations, (4.3) - (4.4) and (4.19) - (4.20), with twenty boundary conditions, (4.11) - (4.13) and (4.67) - (4.70). The objective function for the NLP algorithm is set as

$$F = (\lambda_{\theta_e})^2 \tag{4.223}$$

which is the Earth polar angle costate boundary condition (4.67). The other boundary conditions are enforced by direct equality constraints in the NLP algorithm. The solution of the 2PBVP by two separate methods is done to provide the highest possible accuracy in the final solution. The relaxation method is more robust than the terminal error function method but does not produce as accurate of an answer, i.e., the relaxation solution does not satisfy all of the boundary conditions to a high degree. However, the terminal error method requires a very accurate initial estimate for the unknown initial values which can be provided by the relaxation solution.

The numerical example for a flight time of 8.5 days is given at the end of the MINLP problem description. The number of iterations and CPU times for this problem are given in Table 4.1 and the numerical results section of Chapter 3 provide the "base" solutions convergence properties. These numbers cover the first two steps in the outline above. The relaxation 2PBVP solution for this example required 72 iterations until convergence with an initial defect violation norm of 62.1 which yields an average of 0.1105 for each discrete point where the total number of points was determined by the Hermite-Simpson scheme as 562. The highest violations occur during the translunar coast and usually center between the end of the departure thrust phase and the coordinate transformation point. The terminal error 2PBVP solution required 34 iterations until convergence with an initial constraint norm of 1.2 and one restart. A restart indicates a resetting of the NLP search variables to prevent the nonlinear nature of the problem from disrupting the NLP algorithm. The constraint violation resulted almost entirely on the radial difference at the coordinate cross-over point which had a value of 0.51. The rest of the violation

was spread among the states and costates. The convergence history for this example is "typical" for the majority of the transfers found. The MINLP solutions would often require manual "restarts" and other "tweaks" to converge, but the solution from that point onward was always straightforward and difficulties did not usually arise.

Numerical Results

The solution strategy was used to solve a series of optimal lunar transfer problems for a fixed initial thrust-to-weight ratio of $3\,(10)^{-3}$. The mission profile does not change, i.e., the spacecraft was assumed to start in a circular LEO with an altitude of 300 km and terminate in a circular LLO with an altitude of 100 km and has a fixed initial mass of 100,000 kg. A full engine switching structure was used for various flight times to produce different switching trajectories for both posigrade and retrograde transfers. Figure 4.11 shows the final mass to initial mass performance ratio versus flight time for

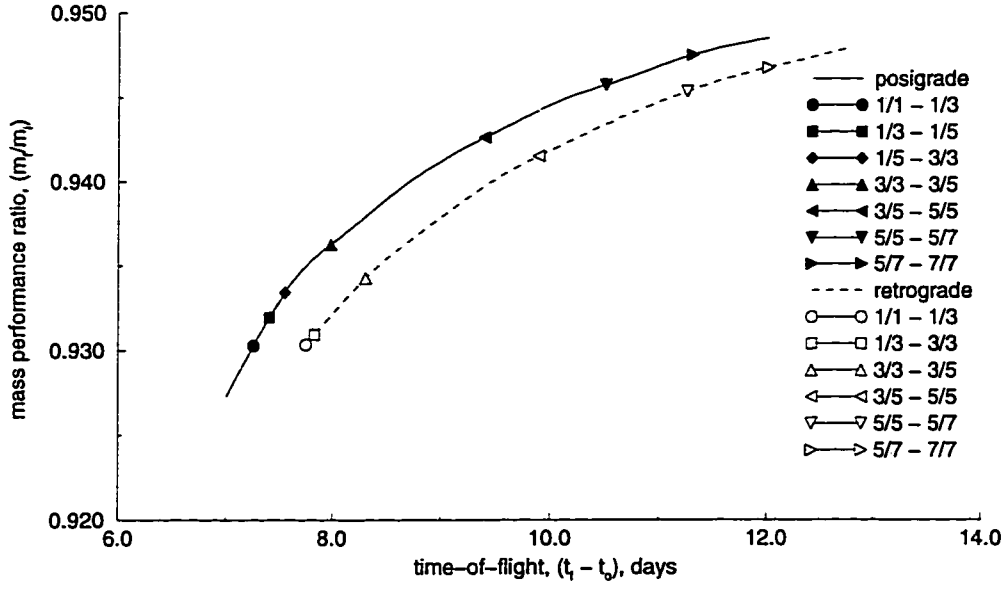

Figure 4.11 Mass Performance Ratio vs Flight Time for Switching Lunar Transfer

both capture types. The total transfer times were varied between 7.0 and 12.0 days for posigrade transfers and 7.75 and 12.25 days for retrograde transfers each in increments of 0.1 days. Both curves have changeover points displayed. These points are approximately the time when the switching structure is altered. The exact point is difficult to specify because of the discontinuous nature of the problem and the large computational expense. The changeover points for the posigrade transfers are denoted by the closed symbols and the retrograde by the open symbols. The legend indicates the type of changeover and consists of two ratios. The first ratio indicates the type of switching structure to the left of the symbol and the second ratio indicates the type of switching structure to the right of the symbol. The ratio indicates the number of switches about the Earth versus the number of switches about the Moon, e.g., a ratio of 3/5 indicates three switches during departure and five switches during capture not counting the initial and final times. The first posigrade changeover point, denoted by a circle, indicates when switching initially takes place and is approximately 7.25 days. The curve to the left of this point has optimal trajectories with two switches, one about the Earth and one about the Moon, as described in Chapter 3. The curve to the right introduces a coast arc about the Moon raising the total number of switches to four. Figure 4.12 shows this type of three-switch posigrade Lunar capture. The second changeover, denoted by a square and occurring around 7.4 days, introduces another coast arc about the Moon raising the total number of switches to six. Figure 4.2 has shown a five-switch posigrade Lunar capture. The third point, denoted by a diamond and occurring around 7.55 days, indicates a new switch about the Earth. However, the total number of switches is still six because the number of switches around the Moon is reduce to three eliminating a coast arc. Figure 4.1 has already shown this type of departure. The fourth point, denoted by a triangle and occurring around 7.98, keeps the three switches about the Earth and adds a new coast arc, or two new switches about the Moon. It is in this region that the example problem of the first section is taken and is shown in Figures 4.1 and 4.2. The fifth changeover

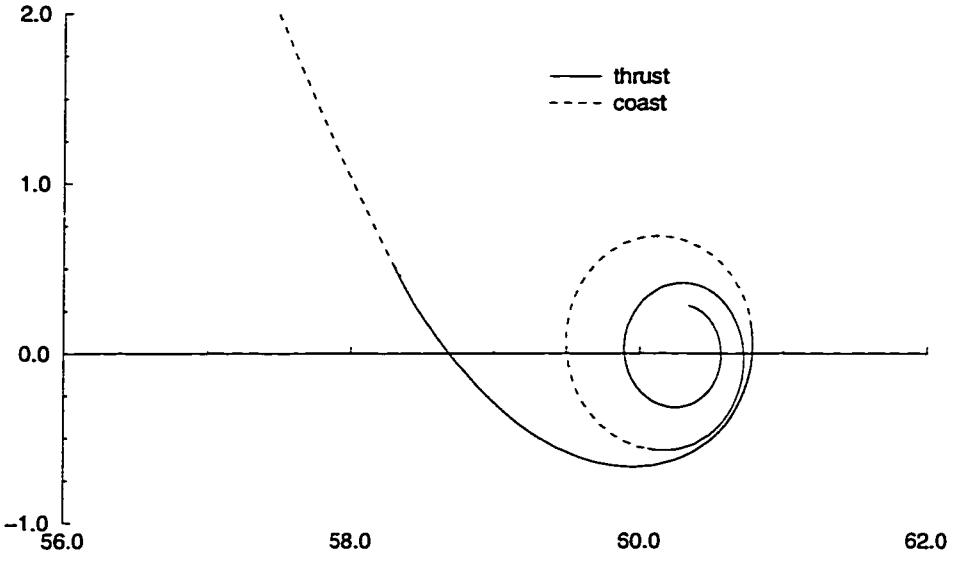

Figure 4.12 Posigrade Lunar Capture Trajectory with 3 Switches

point, denoted by a left-facing triangle and occurring around 8.81 days, adds a new coast arc about the Earth raising the total number of switches to ten. Figure 4.13 shows a Earth departure with five switches. The sixth point, denoted by an upside down triangle and occurring at 10.5 days, adds another coast arc in the Lunar capture raising the total number of switches to twelve. Figure 4.14 shows a posigrade Lunar capture consisting of seven switches. The final point, at 11.3 days, indicates when a new switch occurs about the Earth bringing the total number of switches to fourteen and Figure 4.15 shows this type of departure. The curve begins to level off and it's unlikely that a further switch will occur unless the flight time is increased by a large margin since the seven switch departure allows a wide time latitude for the trajectory. The first coast (three switches) about the Earth occurs after only 0.3 days, the second coast (five switches) at an additional 1.26 days, and the third coast (seven switches) at another 2.49 days. It's unlikely that a an extrapolation of these three points will be very accurate, but an estimate of the next departure switching change would require an additional 3.99

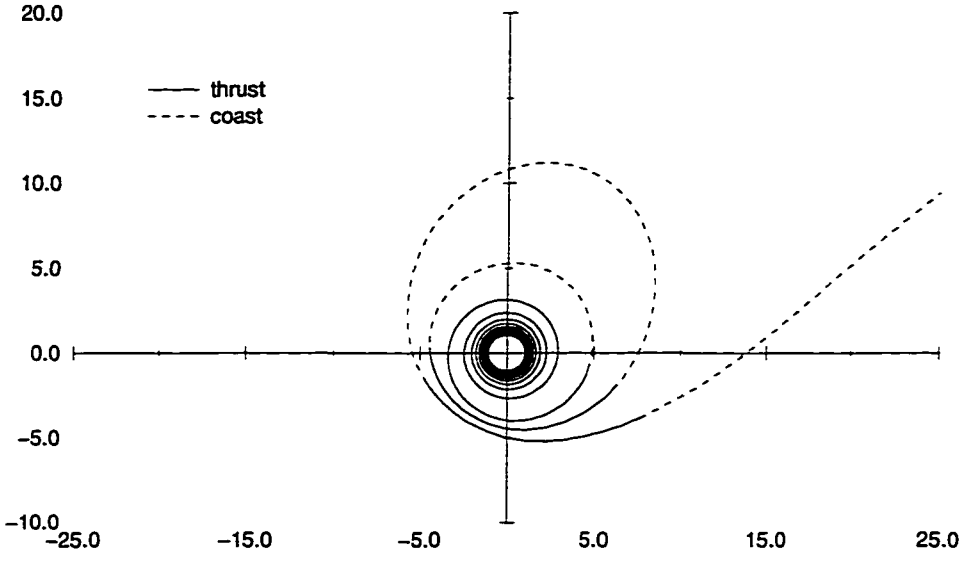

Figure 4.13 Earth Departure Trajectory with 5 Switches

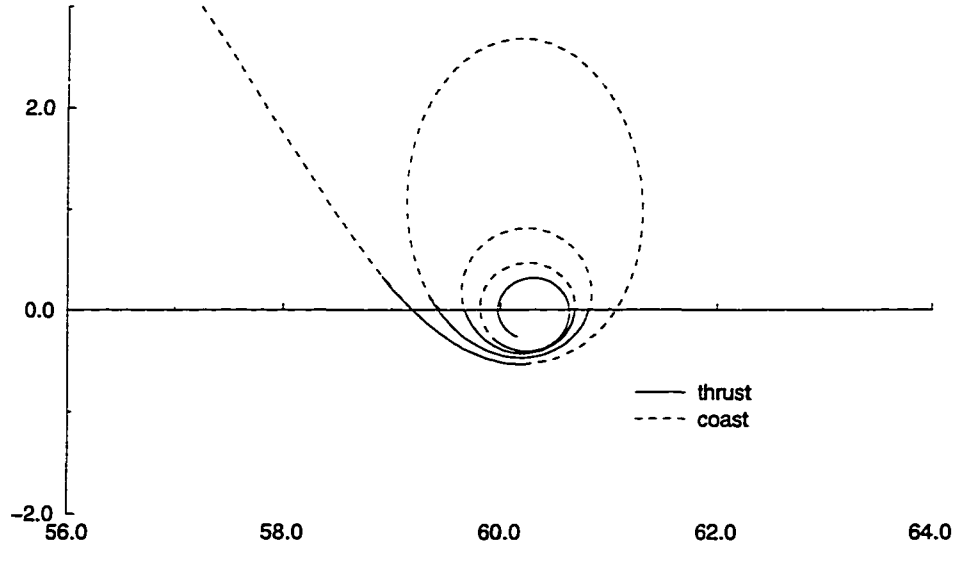

Figure 4.14 Posigrade Lunar Capture Trajectory with 7 Switches

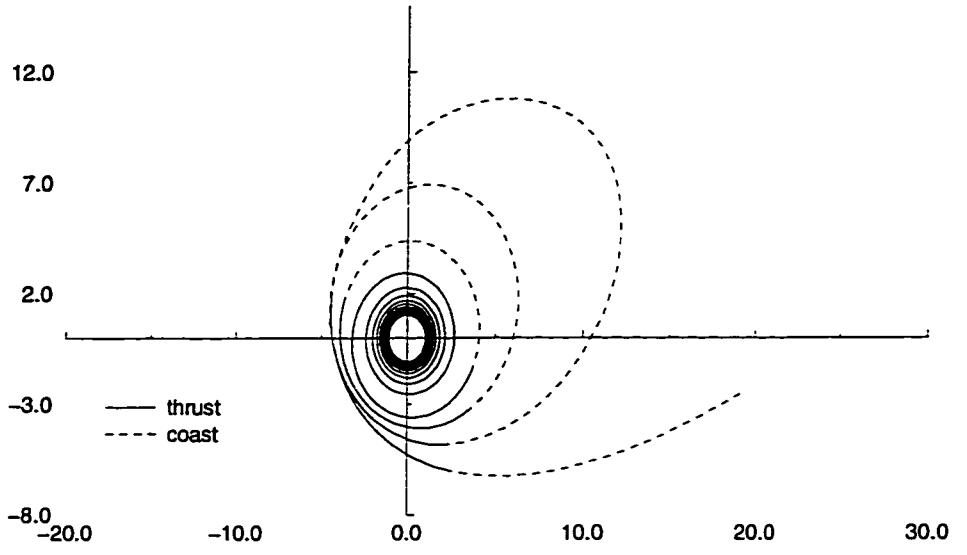

Figure 4.15 Earth Departure Trajectory with 7 Switches

days. Although this is likely conservative, it indicates the probable stability of this final switching structure for the extended performance curve. The retrograde solutions follow the same basic pattern as the posigrade transfers except for the third changeover point on the posigrade curve. The retrograde solutions do not contain a region of six switches, one about the Earth and five about the Moon. The type of symbols for the respective switching structures is kept constant between the curves to clarify this absence. The approximate times of the critical points are 7.75, 7.83, 8.385, 9.4, 11.07, and 12.0 days and follow the same pattern of switching structure as the posigrade solutions with the exception noted above. Figures 4.16 - 4.18 show the retrograde Lunar captures for three, five, and seven switches, respectively.

Several characteristics of the curves and corresponding trajectories can be noted. The major difference between the posigrade and retrograde curves, apart from the time difference for a given performance value, occurs at the lower times-of-flight. The retrograde captures are generally more energy demanding because of the need to overcome

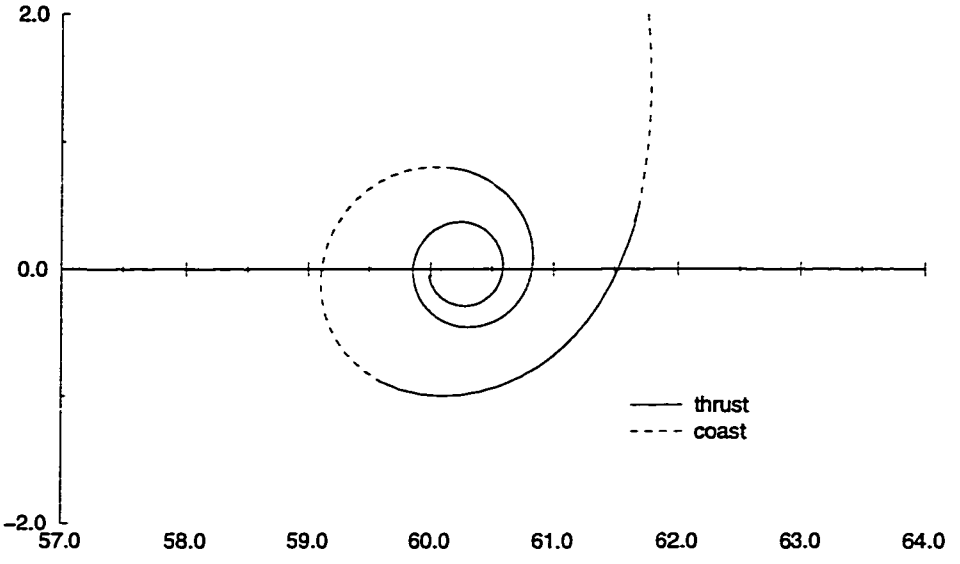

Figure 4.16 Retrograde Lunar Capture Trajectory with 3 Switches

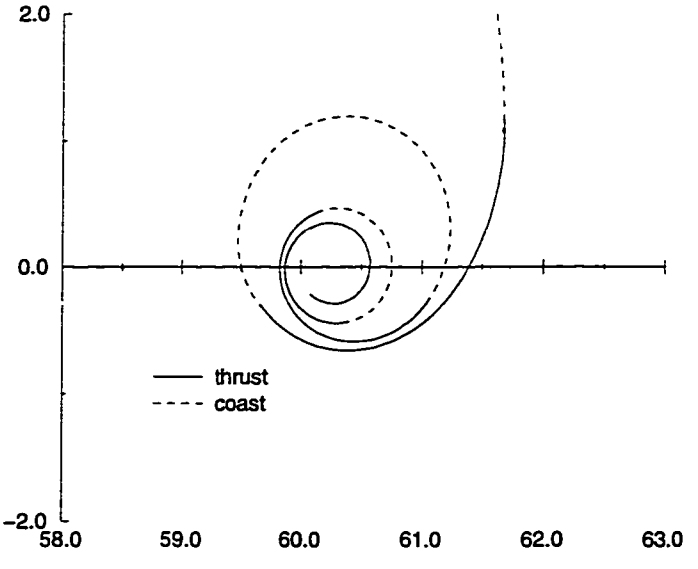

Figure 4.17 Retrograde Lunar Capture Trajectory with 5 Switches

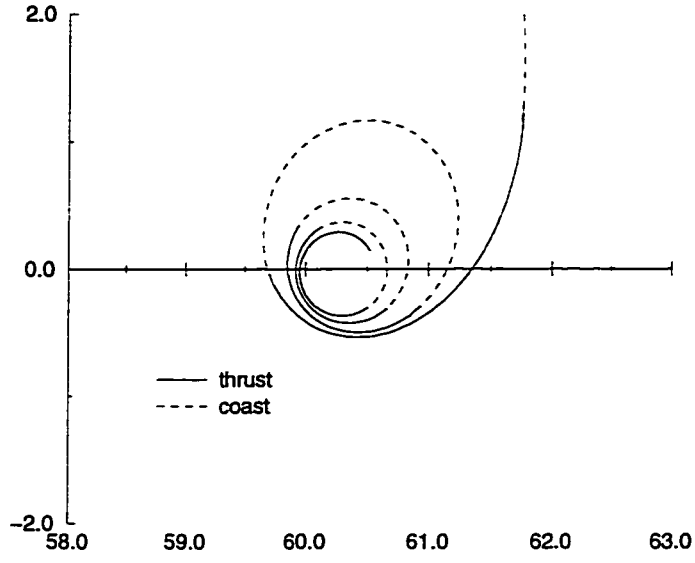

Figure 4.18 Retrograde Lunar Capture Trajectory with 7 Switches

the greater velocity resulting from a retrograde translunar coast. This higher energy demand is exhibited by the thrusting arcs which carry the vehicle farther from the Moon and do not have the same smooth perigee burns shown in the posigrade trajectories at the lower flight times. This higher energy demand also explains the absence of the retrograde curve portion with a ratio of 1/5 since a five switch capture does not allow a large enough energy change in the relatively small time of flight change. The difference between the curves are much smaller at the higher flight times and a similar progression is seen in both performance curves. This is a logical result since the larger flight times place more of the energy change and control burden during the departure phase while the Lunar capture can change the trajectory properties in only a limited way. The extra burden on the departure phase causes the curves to begin to behave in a similar pattern with the difference being the longer translunar coast times needed for the retrograde solutions. Table 4.2 shows a summary of the switching times for the trajectories presented in the figures. The first column gives the figure number for reference. The second

Table 4.2 Departure Switching Times Summary

Departures									
Figure	transfer time days	Switching Times, days							
		1	2	3	4	5	6	7	8
4.1	8.5	0.0	1.64	3.19	3.76				
4.13	9.80	0.0	1.57	1.95	2.21	4.36	4.61		
4.15	11.5	0.0	1.45	1.73	1.92	3.51	3.69	5.94	6.08
Posigrade Captures									
4.12	7.35	6.71	6.94	7.16	7.35				
4.2	8.5	7.69	7.85	8.22	8.28	8.42	8.5		
4.14	10.6	8.91	9.01	10.02	10.13	10.31	10.38	10.49	10.6
Retrograde Captures									
4.16	7.8	7.10	7.31	7.55	7.8				
4.17	9.0	8.08	8.29	8.65	8.78	8.88	9.0		
4.18	11.5	9.99	10.20	11.09	11.18	11.29	11.36	11.43	11.5

column lists the total transfer time for the full transfer. The next two to eight columns give the switching times. The switching times do not sum to the total transfer time because only the characteristic departures and captures are represented. Only one set of departures are given as the retrograde departures do not differ by a large margin from their posigrade equivalents. The number of times given are equal to one more than the number of switches for that specific departure/capture.

CHAPTER 5 NUCLEAR THERMAL ROCKET TRANSFERS

The larger initial thrust-to-weight ratios examined in Chapter 3 require an engine power between conventional chemical rockets and NEP thrusters. The NTR provides this mid-range power requirement but introduces thrust transients that effect the performance of an optimal Lunar mission. The best known studies of NTR's occurred in the late 1960's during the NERVA and KIWI programs, and during the 1980's more studies were done investigating the use of NTR's for the strategic defense initiative (SDI) under the program name TIMBERWIND [8]. These studies either did not account for thrust transients or used a simple mass penalty to estimate the effect on performance. None of the studies attempted to minimize the performance penalty or provided an optimal method of controlling the trajectory or reactor.

Nuclear-Thermal Rocket Modeling

The basic operation of a NTR consists of a gaseous propellant flowing about a nuclear core running at a specified power level. The propellant is heated to a high temperature and then expanded through a nozzle. Figure 5.1 is a schematic of a typical NTR design. The mass flow of propellant is controlled by a pump at the entrance of the core, and so there is a maximum mass flow, i.e., the maximum output of the pump. A reflector is usually included in the NTR design. The reflector, a material which does not absorb stray neutrons or allow their passage, improves the performance of the nuclear core by

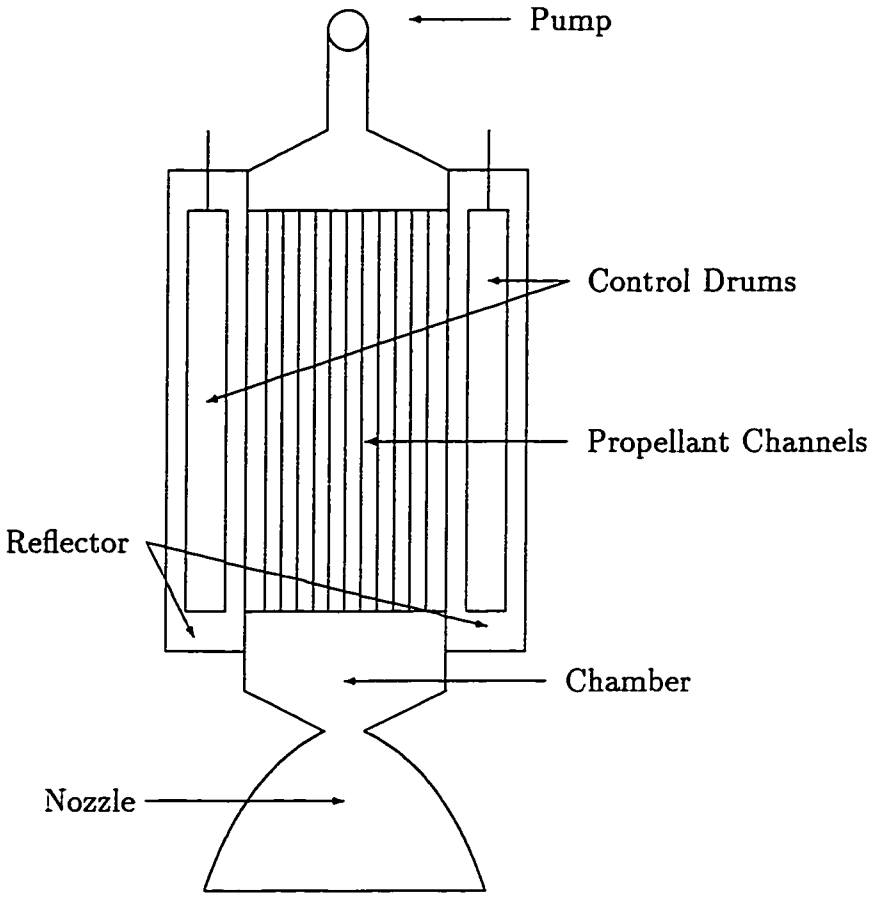

Figure 5.1 Schematic Diagram of NTR

keeping the fissioning neutrons inside and preventing die-off or power losses along the outer region. The method of control consists of drums or vanes placed in the reflector which control the reactivity by presenting either a reflector surface or large neutron-capture material to the interior of the core. The drums are rotated and are capable of extremely rapid control response, usually on the order of milliseconds.

The propellant is heated by the core and cools the core at the same time. The thrust transient effects occur at both startup and shutdown of the NTR. The startup transient is caused by the need for the rocket to change rapidly from a low-level steady-state power output where the mass flow is some minimum to a high-level steady-state power output where the nominal thrust values for the NTR are achieved. This occurs relatively

rapidly and the entire transient from low-power to high-power typically required 10-20 seconds for the NERVA/KIWI engines. The shutdown transient is a far more serious consideration. After shutdown has occurred, i.e., the reactivity in the core is such that the new neutron fissions are dying off, heat is still generated by the effect of delayed neutrons. The reactor would quickly exceed the maximum allowable temperature unless cooling is continued by the propellant. In this chapter, I attempt to use this cooling period in an optimal fashion to obtain useful thrust.

Reactor Core Dynamics

The fissioning of nuclear fuels, such as U^{235}, is done by neutrons and immediately produces γ-radiation, high speed fission fragments, and between two and three high speed or fast neutrons. The majority of the heating effects comes from the kinetic energy of the fragments or fission products. The fission products themselves are radioactive, and their decay leads to a further energy release for some time after the fission process has stopped. This phenomenon of delayed neutrons is the primary cause of the thrust transient effects. The mono-energetic neutron kinetic equations assume that the core can be simulated by a point energy source and ignores any particular core geometry or fuel placement. A more advanced analysis can be done by forming the partial differential neutron flux equations which model the core in time and space. This advanced simulation is not necessary for this work, since the extra complication would not greatly effect the general solution. Also, the NERVA/KIWI programs used the mono-energetic equations for their modeling and they proved sufficient for preliminary core design. The mono-energetic neutron kinetics [31] are described by the linear dynamics

$$\frac{dn}{dt} = \frac{\rho - \gamma}{\Lambda} n + \sum_{i=1}^{6} \delta_i \bar{c}_i \tag{5.1}$$

$$\frac{d\bar{c}_i}{dt} = \frac{\gamma_i}{\Lambda} n - \delta_i \bar{c}_i \qquad i = 1, \ldots, 6 \tag{5.2}$$

where

n = number of neutrons in core

$\bar{c}_i$ = number of precursor neutrons of the i type

ρ = reactivity

γ = total delayed neutron fraction = $\sum \gamma_i$

Λ = neutron reproduction time

δ_i = mean decay rate of precursors of the i type

γ_i = fraction of delayed neutrons for the i type

The reactor's power is moderated or controlled by specifying the reactivity ρ, which is a measure of the rate of neutron production. This rate can be modified by changing the absorption properties of the core, i.e., how quickly the free neutrons are allowed to produce more free neutrons. The constants γ_i and δ_i and the number of precursor groups are specific to the type of nuclear fuel used, and Table 5.1 shows the values for U^{235}.

Table 5.1 Group Constants for U^{235}

group	γ_i relative yield	δ_i decay rate, s^{-1}
1	0.000266	0.0127
2	0.001492	0.0317
3	0.001317	0.1150
4	0.002851	0.3110
5	0.000897	1.4000
6	0.000182	3.8700

The total delayed neutron fraction is the sum of the delayed fractions for each group and has a value of $\gamma = 0.007005$.

The neutron reproduction time is a function of the core design, type of fuel used, placement of that fuel, void spaces, etc. Fortunately, Λ is nearly constant for a given

design and can be approximated by the power level of the reactor and whether the reactor uses a reflector or not.

The heat generated by the core is proportional to the number of neutrons, i.e.,

$$q = K_q n \tag{5.3}$$

where q is the heat generated. Equations (5.1) - (5.2) can be rewritten in terms of q,

$$\frac{dq}{dt} = \frac{\rho - \gamma}{\Lambda} q + \sum_{i=1}^{6} \delta_i c_i \tag{5.4}$$

$$\frac{dc_i}{dt} = \frac{\gamma_i}{\Lambda} q - \delta_i c_i \qquad i = 1, \ldots, 6 \tag{5.5}$$

where c_i is now in units of power. Equation (5.4) now represents the current power output of the core. The steady state values for delayed neutron precursor groups can be written in terms of the current power setting of the reactor,

$$c_{io} = \frac{\gamma_i}{\delta_i} \frac{q_o}{\Lambda} \tag{5.6}$$

The boundary conditions for a low-power startup could then be written as

$$q(t_o) = q_o \tag{5.7}$$

$$c_{io} = \frac{\gamma_i}{\delta_i} \frac{q(t_o)}{\Lambda} \tag{5.8}$$

where q_o is the low-power steady-state operation of the core. The shutdown conditions are identical except q_o is replace by some desired maximum power-output.

The reactivity of the core is the control used to change the power level. It is usually bounded, but a more realistic bound would be the rate at which the reactivity changes. The reactivity ρ can be treated as a state, and its state equation can be written as:

$$\frac{d\rho}{dt} = u_\rho - \alpha_\rho T_c \tag{5.9}$$

where u_ρ is the new control which is almost always bounded, i.e., $|u_\rho| \leq u_\rho^{\max}$, and T_c is the current core temperature defined later in this section. The limit on reactivity

change can be seen as a measure of how the reactivity process operates. The absorption properties of a core determine its reactivity and this absorption can be modified by control vanes/drums. Neutrons near to the vanes/drums react to the change in control immediately and neutrons farther away react less quickly. The extra time necessary for the control action to effect enough of the free neutron population to change the reactivity can be approximated by the reactivity rate bound $u_\rho^{\max}$. The variable α_ρ is called the coefficient of temperature on reactivity and is dependent on the core design. A typical value of α_ρ is $2(10)^{-5}\,K^{-1}$, and this represents a stabilizing effect on the reactor. A higher core temperature causes the reactivity to decrease lowering the power output and lowering the heat produced. Introducing the reactivity as a state variable makes Equation (5.4) nonlinear rather than bilinear when ρ is a control variable.

Heat-Exchange Model

A single-lump heat exchange model is used to estimate the temperature change for propellant through the core as well as the core temperature itself. A simple energy balance [32] provides the following equations:

$$\frac{dT_c}{dt} = \frac{1}{c_c}\left[q - hA\left(T_c - T_p\right)\right] \tag{5.10}$$

$$\frac{dT_p}{dt} = \frac{1}{c_p}\left[hA\left(T_c - T_p\right) - c_m\beta\left(T_o - T_i\right)\right] \tag{5.11}$$

where

T_c = temperature of the core, K

c_c = the reactor core heat capacity, J/K

h = the heat transfer coefficient

A = a representative area of heat transfer, m^2

T_p = temperature of the propellant in the core, K

c_p = the propellant mass heat capacity, J/K

c_m = the propellant specific heat $\approx 1008\ \ J/(kg)(K)$ [32]

T_o = the outlet temperature of the propellant from the core, K

T_i = the inlet temperature of the propellant into the core, K

The reactor core heat capacity, c_c, is a function of the type of fuel, moderator material, and engine dimension. The heat transfer coefficient, h, can be approximated by a constant as suggested by Glasstone and Sesonske [33]. This constant can be determined from the nominal value of the mass flow at the rated power level of the rocket, and is given by

$$h = C_h\,(\dot{m}_{nom})^{4/5} \tag{5.12}$$

where $\dot{m}_{nom}$ is the nominal mass flow at the rated power level , i.e., β_{max}. The coefficient is given by

$$C_h = C_n n^{-0.8} d^{-1.8} \tag{5.13}$$

$$C_n = 0.462\,(10)^{-3}\,\frac{kJ}{(msec)^{0.2}\,K\,(kg)^{0.8}} \tag{5.14}$$

where n and d are the number of channels and their diameters. The assumption of a constant heat transfer coefficient does not seriously degrade the accuracy of the transient modeling but can effect the long-term modeling of the NTR. The heat transfer coefficient is used to determine the useful lifetime of a reactor, and the constant assumption will invalidate any projections. Since this is not a concern here, the assumption is used. The representative heat transfer area, A, depends on how the propellant is heated, i.e., the geometry of the propellant channels. The typical engine uses cylindrical cooling channels so that the representative heating area becomes

$$A = n\,(\pi\ \ d\ \ l) \tag{5.15}$$

where l is the length of the channel and n and d are the same as before. The propellant mass heat capacity is a function of the specific heat and the current amount of propellant flow in the reactor.

The inlet temperature T_i of the coolant is usually fixed for a given NTR design, and the relation between T_p and T_o and T_i is given by [34]

$$T_p = \frac{T_i + \phi T_o}{1 + \phi} \tag{5.16}$$

where $1 \leq \phi \leq 2$. The variation of ϕ allows a primitive form of estimating the power distribution in the core, where value of $\phi = 1$ is a constant power level.

Rocket-Nozzle Performance

The thrust of a rocket is defined by [32]

$$T = \dot{m}u_e + (p_e - p_\infty) = \dot{m}C \tag{5.17}$$

were u_e is the exhaust velocity, p_e is the exit pressure, p_∞ is the ambient pressure, and C is the effective exhaust velocity. The rocket is assumed to always operate in space so the ambient pressure can be neglected, and the effective exhaust velocity can then be written as:

$$C = u_e + \frac{p_e A_e}{\dot{m}} \tag{5.18}$$

The exhaust velocity [32] can be written as

$$u_e = \sqrt{\frac{2\tilde{\gamma}}{\tilde{\gamma}-1}}\left(\frac{2}{\tilde{\gamma}+1}\right)^{(\tilde{\gamma}+1)/2(\tilde{\gamma}-1)}\sqrt{\tilde{\gamma}}C^*\left[1-\left(\frac{p_e}{p_c}\right)^{(\tilde{\gamma}-1)/\tilde{\gamma}}\right]^{1/2} \tag{5.19}$$

where C^* is the characteristic velocity, and p_e/p_c is the pressure ratio between the nozzle exit and nozzle, and $\tilde{\gamma}$ is the gas constant. The characteristic velocity can also be written in terms of the nozzle inlet stagnation temperature as

$$C^* = \left(\frac{\tilde{\gamma}+1}{2}\right)^{(\tilde{\gamma}+1)/2(\tilde{\gamma}-1)}\sqrt{\frac{RT_o}{\tilde{\gamma}}} \tag{5.20}$$

The effective exhaust velocity can be written by combining these terms,

$$C = \left(\frac{2\tilde{\gamma}}{\tilde{\gamma}-1}RT_o\right)^{1/2}\left[1-\left(\frac{p_e}{p_c}\right)^{(\tilde{\gamma}-1)/\tilde{\gamma}}\right]^{1/2} + \frac{p_e A_e}{\dot{m}} \tag{5.21}$$

The exit pressure is usually very small, and neglecting those terms yields the effective exhaust velocity in terms of the inlet nozzle temperature,

$$C \approx \sqrt{\frac{2\tilde{\gamma}}{\tilde{\gamma}-1}RT_o} \tag{5.22}$$

Equations (5.17) and (5.22) link the temperature of the propellant leaving the core to the effective thrust given to the spacecraft.

Problem Statement and Discussion

The development of a minimum-fuel Lunar transfer with thrust transient effects combines the dynamic equations for the three-body Earth-Moon system, the mono-energetic neutron equations, and the lumped-mass heat exchange modeling. Two separate objectives are being treated here. The first is to use the least amount of fuel to accomplish the transfer, which has been been dealt with in preceding chapters, and the second is to use the least amount of control effort to regulate the reactor. The minimum reactor control effort goal is enforced by adding an integral term in the objective function given in the problem statement. The new integral term raises the question of whether this type of transfer is still close to a minimum-fuel trajectory. An analysis of the resulting optimal control equations and numerical results given below show that the solutions to the problem essentially provide a minimum-fuel transfer. The problem statement for the coupled system is given by:

Find $u(t)$, $\beta(t)$, and $u_\rho(t)$ for the time interval $t_o \leq t \leq t_f$ which minimize

$$J = -m(t_f) + \int_{t_o}^{t_f} |u_\rho|\, dt \tag{5.23}$$

subject to

$$\dot{r} = V_r \tag{5.24}$$

$$\dot{\theta} = \frac{V_\theta}{r} \tag{5.25}$$

$$\dot{V}_r = \frac{V_\theta^2}{r} - \frac{\mu_e}{r^2} + \mu_m \cos\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - \mu_m \frac{r}{r_m^3} + \omega^2 r \tag{5.26}$$

$$+2\omega V_\theta + \frac{C\beta}{m} \sin u \tag{5.27}$$

$$\dot{V}_\theta = -\frac{V_r V_\theta}{r} - \mu_m \sin\theta \left(\frac{d}{r_m^2} - \frac{1}{d^2} \right) - 2\omega V_r + \frac{C\beta}{m} \cos u \tag{5.28}$$

$$\dot{m} = -\beta \tag{5.29}$$

$$\dot{q} = \frac{\rho - \gamma}{\Lambda} q + \sum_{i=1}^{6} \delta_i c_i \tag{5.30}$$

$$\dot{c}_i = \frac{\gamma_i}{\Lambda} q - \delta_i c_i \qquad i = 1, \ldots, 6 \tag{5.31}$$

$$\dot{\rho} = u_\rho - \alpha_\rho T_c \tag{5.32}$$

$$\dot{T}_c = \frac{1}{c_c} \left[q - hA \left(T_c - T_p \right) \right] \tag{5.33}$$

$$\dot{T}_p = \frac{1}{c_p} \left[hA \left(T_c - T_p \right) - c_m \beta \left(T_o - T_i \right) \right] \tag{5.34}$$

satisfying the boundary conditions $\psi \left[x \left(t_o \right), x \left(t_f \right) \right] = 0$ where

$$\psi_1 = r \left(t_o \right) - r_{LEO} \tag{5.35}$$

$$\psi_2 = V_r \left(t_o \right) \tag{5.36}$$

$$\psi_3 = V_\theta \left(t_o \right) - \sqrt{\mu_e / r_{LEO}} + \omega r_{LEO} \tag{5.37}$$

$$\psi_4 = m \left(t_o \right) - m_{LEO} \tag{5.38}$$

$$\psi_5 = q \left(t_o \right) - q_o \tag{5.39}$$

$$\psi_j = c_i \left(t_o \right) - \frac{\gamma_i q_o}{\delta_i \Lambda} \qquad j = 6, \ldots, 11 \qquad i = j - 5 \tag{5.40}$$

$$\psi_{12} = \rho \left(t_o \right) \tag{5.41}$$

$$\psi_{13} = T_c \left(t_o \right) - T_{c_o} \tag{5.42}$$

$$\psi_{14} = T_p \left(t_o \right) - T_i \tag{5.43}$$

$$\psi_{15} = r_m \left(t_f \right) - r_{LLO} \tag{5.44}$$

$$\psi_{16} = V_{r_m}(t_f) \tag{5.45}$$

$$\psi_{17} = V_{\theta_m}(t_f) - \sqrt{\mu_m / r_{LLO}} + \omega r_{LLO} \tag{5.46}$$

$$\psi_{18} = q(t_f) - q_f \tag{5.47}$$

$$\psi_j = c_i(t_f) - \frac{\gamma_i q_f}{\delta_i \Lambda} \qquad j = 19, \ldots, 24 \quad i = j - 18 \tag{5.48}$$

$$\psi_{25} = \rho(t_f) \tag{5.49}$$

$$\psi_{26} = T_c(t_f) - T_{c_f} \tag{5.50}$$

and the control limits of $0 < \beta_{\min} \leq \beta(t) \leq \beta_{\max}$ and $|u_\rho| \leq u_\rho^{\max}$. The combined system consists of a fixed end-time problem with fifteen states, three controls of which two are bounded, fourteen initial state constraints, and twelve final state constraints. Equations (5.24) - (5.29) are identical to the dynamic equations described earlier in Chapters 3 and 4 for the restricted three-body problem. Equations (5.30) - (5.32) are the mono-energetic power equations with reactivity control described earlier, and Equations (5.33) - (5.34) are the lumped-mass heat exchange dynamics. The three systems are coupled through the exhaust velocity C which is proportional to the square root of the propellant temperature minus a constant via Equations (5.16) and (5.22). Another coupling occurs in Equation (5.34) where the massflow is used to determine the change in propellant temperature change. Two of the control histories are bounded. The mass flow boundaries have changed slightly from Chapter 4 in that the lower bound is now non-zero. This is necessary since the core will always require some cooling for the structural elements, although the value can be quite small. The initial and terminal state constraints are a combination of both trajectory, core, and heat-exchanger information. ψ_1 through ψ_4 and ψ_{15} through ψ_{17} are the trajectory boundary conditions specifying the initial circular LEO and final circular LLO. ψ_5 and ψ_{18} specify the initial and final power rating for the reactor core, and ψ_6-ψ_{11} and ψ_{19}-ψ_{24} require the core to be in a steady-state at both times. ψ_{13} and ψ_{26} specify the initial and final core temperatures, and ψ_{14} is the propellant temperature immediately at start-up. ψ_{12} and ψ_{25} specify a zero reactivity

at the boundaries which ties in with the steady-state condition.

The Hamiltonian for this system is

$$H = |u_\rho| + \lambda_r(\dot{r}) + \lambda_\theta\left(\dot{\theta}\right) + \lambda_{V_r}\left(\dot{V}_r\right) + \lambda_{V_\theta}\left(\dot{V}_\theta\right) + \lambda_m(\dot{m}) \tag{5.51}$$

$$+\lambda_q(\dot{q}) + \sum_{i=1}^{6}\lambda_{c_i}(\dot{c}_i) + \lambda_\rho(\dot{\rho}) + \lambda_{T_c}\left(\dot{T}_c\right) + \lambda_{T_p}\left(\dot{T}_p\right) \tag{5.52}$$

The costate system for the first five costates, λ_r, λ_θ, λ_{V_r}, λ_{V_θ}, λ_m, is identical to those derived in Chapter 3 and will not be repeated here. The rest of the costate system is

$$\dot{\lambda}_q = \frac{\gamma-\rho}{\Lambda}\lambda_q - \frac{1}{\Lambda}\sum_{i=1}^{6}\gamma_i\lambda_{c_i} - \frac{\lambda_{T_c}}{c_c} \tag{5.53}$$

$$\lambda_{c_i} = \delta_i(\lambda_{c_i} - \lambda_q) \qquad i = 1,\ldots,6 \tag{5.54}$$

$$\dot{\lambda}_\rho = -\frac{q}{\Lambda}\lambda_q \tag{5.55}$$

$$\dot{\lambda}_{T_c} = hA\left(\frac{\lambda_{T_c}}{c_c} - \frac{\lambda_{T_p}}{c_p}\right) + \lambda_\rho\alpha_\rho \tag{5.56}$$

$$\dot{\lambda}_{T_p} = -\frac{\beta}{m}\left(\frac{\tilde{\gamma}}{\tilde{\gamma}-1}\right)\left(\frac{\phi+1}{\phi}\right)(\lambda_{V_r}\sin u + \lambda_{V_\theta}\cos u)\frac{R}{C} \tag{5.57}$$

$$-hA\left(\frac{\lambda_{T_c}}{c_c} - \frac{\lambda_{T_p}}{c_p}\right) + \frac{c_g}{c_p}\frac{(1+\phi)}{\phi}\beta\lambda_{T_p} \tag{5.58}$$

The mass flow control once again yields a switching function because it appears linearly in the dynamic equations. The switching function is similar to the non-transient case and is given by

$$\beta^* = \begin{cases} \beta_{\max}, & \hat{\sigma} < 0 \\ 0, & \hat{\sigma} > 0 \end{cases} \tag{5.59}$$

where

$$\hat{\sigma} = \frac{C}{m}(\lambda_{V_r}\sin u + \lambda_{V_\theta}\cos u) - \lambda_m - \frac{c_m(T_o - T_i)}{c_p}\lambda_{T_p} \tag{5.60}$$

Equation (5.60) is the same as the earlier switching functions but includes a new term accounting for the propellant temperature costate and nozzle inlet temperature. The reactivity rate control also has a switching function but its nature is different because of the integral term in the performance index. The Pontryagin Minimum Principle (PMP)

requires that the Hamiltonian be minimized over all admissible control for optimal states and costates, which is expressed in equation form as

$$H(r^*, \lambda^*, u^*, t) \leq H(r^*, \lambda^*, u, t) \tag{5.61}$$

Applying the PMP to the above problem yields

$$\left|u_\rho^*\right| + \lambda_\rho^* u_\rho^* \leq |u_\rho| + \lambda_\rho^* u_\rho \tag{5.62}$$

which is satisfied for a "bang-off-bang" control for u_ρ of the form

$$u_\rho^* = \begin{cases} u_\rho^{\max}, & \lambda_\rho < -1 \\ 0, & -1 < \lambda_\rho < 1 \\ -u_\rho^{\max}, & \lambda_\rho > 1 \end{cases} \tag{5.63}$$

The switching will then depend upon the current power of the reactor q and the corresponding costate Equation (5.55). The time history of the reactivity and mass flow controls follow general patterns for all engine starts and shutdowns. The startup of the reactor increases the power level to some maximum using the least amount of propellant. This increase in power, or core temperature, can be accomplished by either decreasing the mass flow to its smallest value and waiting for the core to naturally heat, or the reactivity can be increased by increasing the reactivity rate. It's easily apparent that increasing the reactivity will use less propellant and bring the reactor to power in a much smaller time. Once the reactor is near maximum power, the mass flow is increased to its maximum and the nominal thrust is produced. The same logic leads to the general shutdown strategy. The mass flow is left at its maximum while the reactivity is lowered at the fastest possible rate and, near the lower power limit, the mass flow is switched to the lower value so that critical cooling of structural elements can continue. The steady-state cases where the reactor is run at $q^{\max}$ or $q^{\min}$ for extended periods can be accomplished by having the reactivity costate λ_ρ lie between -1 and 1 which allows the core to remain at a steady state. Earlier the question was raised of whether this

form of control still yields a minimum-fuel transfer with the altered performance index. The only effect of this new term is to change the form of the reactivity control and the costate equations and mass flow switching function are unaltered. A formulation of the problem without the integral term gives another switching function for the reactivity rate and has the form,

$$u_\rho^* = \begin{cases} u_\rho^{\max}, & \lambda_\rho < 0 \\ -u_\rho^{\max}, & \lambda_\rho > 0 \end{cases} \tag{5.64}$$

This type of switching function would require the reactivity rate to rapidly oscillate during any steady-state operation in order for the reactor to remain at an equilibrium condition. The implementation of this type of rapidly varying control is possible but in practice a "zero" control is enforced to eliminate the need. The inclusion of the integral term in the performance index is done primarily to provide this "zero" control for the steady-state operation of the core and still provide a very-nearly minimal-fuel trajectory.

Three different state history constraints need to be imposed to secure both the structural integrity of the reactor and limit the physical parameters. The first is a maximum power limit which prevents the reactor from a "melt down" condition. The power produced is limited by

$$q(t) \leq q^{\max} \tag{5.65}$$

where $q^{\max}$ is the rated power level of the reactor. This limit could also be imposed by setting a maximum temperature for the core, but this approach does not ease the solution or change the solution for a given mission profile. The second state history constraint is imposed upon the rate of change of the reactor temperature and is necessary to prevent thermal stress fractures. A full analysis of the core physical geometry and neutron flux is necessary to accurately limit the temperature change, but a simple bound is sufficient for this work and takes the form

$$\frac{dT_c}{dt} \leq 1700 K/sec \tag{5.66}$$

where the values are taken from the suggestions by both Mohler and Shen [34] and the Rover base report [6] developed at NASA Lewis NTR Workshop of the early 1990's. The third and last state history constraint limits the total reactivity to a maximum value ρ^{max}. This limit on reactivity imposes the physical limitations on the reactor and further prevents any "melt down" condition as described for the power constraint above. The inclusion of these constraints is difficult to implement for any terminal error function approach but is possible with the collocation method detailed in Chapter 2 and implemented for switching trajectories in Chapter 4. A new series of inequality constraints are introduced which bound the current power level and the temperature time derivative at every discrete point. Because of the algebraic relations linking the discrete points together to form the trajectory, we are not assured of having q and dT_c/dt bounded in between the points. This problem can be alleviated by slightly reducing the limit or increasing the number of points in those areas where the boundary is thought to be exceeded.

The effects of the thrust transients will be accounted for in two different cases. The first case occurs where there is only one engine "burn" and the thrust arc is essentially isolated from the rest of the trajectory. This case essentially takes the reactor from low-power and low-core temperature to the engine's rated power and maximum allowable core temperature, sustains this power level for a duration determined by the needs of the translunar trajectory, and then lowers both the power and temperature to their lower steady-state values. The behavior of the control can then be deduced from the above discussion where the beginning of the "burn" causes a startup and the termination of the "burn" causes a shutdown. The second transient case involves the partial cooling and then reheating of a reactor because the needed thrust phases are close together or the thrust arcs are not isolated. This second case is one of the reasons that the reactor core and lumped-mass heat exchange models were developed and little can be said of what the reactor and reactivity time histories will be.

Numerical Results

The solution of a particular thrust transient Lunar transfer involves both trajectory and reactor information. The trajectory information is presented first since its format is similar to data given in the preceding chapters. The reactor core results are then presented with a discussion on any unique characteristics. The transient solution is found and a comparison is given between the non-transient case and other propulsion systems and transient modeling. The effects of transients are studied for a specific transportation mission profile suggested by Borowski [35], [9]. The mission profile has changed from those set for the previous chapters. The spacecraft now starts in a 485 km circular LEO and terminates in a 115 circular LLO. The total flight time is set at 5.25 days. These orbits and flight time are set to provide a sequence of vehicle locations with unobstructed access to a Lunar outpost which is the basis for the NTR use. A description of the mission profile, the outpost, and the reasons for the necessary requirements are given by Borowski [9].

The maximum chamber temperature is set at $T_c^{\max} = 3500$ K which yields a specific impulse of $I_{sp} = 1,100$ seconds. The propellant core inlet temperature is set at $T_i = 70$ K as suggested by [6] and [34]. The maximum engine power is set at $q^{\max} = 1500$ MW with a rated thrust level of $T = 333.6$ kN, and the minimum engine power is set at $q^{\min} = 0.01$ MW. The maximum mass flow is then found to be $\beta_{\max} = 30.925$ kg/sec and the minimal mass flow is set at $\beta_{\min} = 1.2\,(10)^{-2}$ kg/sec which is calculated from the maximum power by the method suggested by the NASA Lewis Rover/NTR Report [6]. The reactivity rate limit is set at $u_\rho^{\max} = 0.003$ sec^{-1} and reactivity is limited to $\rho^{\max} = 0.9\gamma$. The engine itself uses the hexagonally-shaped fuel elements developed for the original NERVA reactors each of which is $l = 1.3208$ m long, contains 19 axial coolant channels and produces 1.4 megawatts of power. Thus, 1250 elements are needed and the total engine contains $n = 23,750$ propellant channels. The channel diameter is

set at $d = 4.0$ mm which produces a L/D ratio of 330.2 which is technically possible according to Oates [32], [6], and Watson [36]. The density of graphite moderated U^{235} is 1617.9 kg/m^3 with a specific heat of 1.966 kJ/(kg)(K). The core heat capacity can be approximated by knowing the density, specific heat, and number of fuel elements and is found to be 1.0. The neutron reproduction time, Λ, is set at $\Lambda = 1.4(10)^{-4}$ which is determined by the power level and the hexagonal fuel elements. The mass in LEO is set at $3.4(10)^5$ kg which results in an initial thrust-to-weight ratio of $(T/W)_i = 0.1$. This amount is specified to allow a brief comparison of the solution with other transient models as given by Zimmerman [37] and Rivas [38].

Initial Estimates

The combination of several systems in the thrust transient problem require the patching of several different initial estimates. The spacecraft position and velocity data was estimated by solving the non-transient case for the same flight time and $(T/W)_i$ using the methods described in Chapter 4 for a fixed flight time switching trajectory. The MINLP approximation converged in 23 iterations, the relaxation 2PBVP solution converged in 12 iterations, and the final terminal error 2PBVP solution converged in 18 iterations. The final mass for the non-transient case was found to be 239,771 kg. This non-transient solution provides estimates for the times histories for Equations (5.24) - (5.29) and allows estimates of the number and timing of the engine switches.

The reactor and heat-exchange dynamics were estimated for each thrust phase from the solutions to two simpler subproblems of a core startup and shutdown. The startup of the reactor can be approximated by solving the subproblem:
Find $\beta(t)$ and $u_\rho(t)$, $0 \leq t \leq 1$, and α which minimize:

$$J = \alpha + \int_0^1 |u_\rho| \, dt \tag{5.67}$$

subject to Equations (5.30) - (5.34) with the initial conditions

$$q = q_o \tag{5.68}$$

$$c_1 = \frac{\gamma_1 q_o}{\delta_1 \Lambda} \tag{5.69}$$

$$\vdots = \vdots \tag{5.70}$$

$$c_6 = \frac{\gamma_6 q_o}{\delta_6 \Lambda} \tag{5.71}$$

$$\rho = 0 \tag{5.72}$$

and satisfying $\Psi = 0$ where

$$\psi_1 = q(t_f) - q_f \tag{5.73}$$

$$\psi_j = c_i(t_f) - \frac{\gamma_i q_f}{\delta_i \Lambda} \qquad j = 2, \ldots, 7 \quad i = j - 1 \tag{5.74}$$

$$\psi_8 = \rho \tag{5.75}$$

and where $0 < \beta_{\min} \leq \beta(t) \leq \beta_{\max}$ and $|u_\rho| \leq u_\rho^{\max}$. This subproblem consists of seven states and seven terminal state constraints and two bounded controls. The time control parameter α is used to find the minimum time required to bring the reactor to full power which closely approximates a minimum-fuel startup. The solution to this subproblem gives initial estimates for the reactor and heat-exchange dynamics for all startups of each thrust phase. The shutdown estimates are solved by a similar problem but with different boundary conditions. Rather than go from a low-power $q = q_o$ to a high-power $q = q_f$, the shutdown reverses the boundaries to go from q_f to q_o in a minimum time. The solution to both of these problems are straightforward, and initial core designs employ their solution so a wealth of data is available on their development [39], [33], [36]. The times spent during the steady-state are simply set at constants for their respective phases, i.e., the coast phases are at low-power and the thrust phases are at hi-power. The solution to the subproblems and the constant steady-state estimates provide the time history estimates for Equations (5.30) - (5.34).

Solution for a 5.25 day Transfer

The relaxation method for the solution of a 2PBVP was used for the solution of the transfer because of the state variable constraints on temperature and power and the limit on the rate of core temperature change. The total number of discrete points was set at 850 points and left constant to insure accuracy of any of the unknown costate properties. The points were distributed manually rather than use a automatic distribution for simplicity. One hundred fifty points are distributed during the Earth departure, two hundred fifty for the translunar coast, and four hundred points for the Lunar capture. The Lunar capture will introduce any unique control characteristics so the points were distributed to account for this. The magnitude of the initial error residual vector resulting from the estimates described above was 82.1 where 62% of the error magnitude is attributable to the Lunar capture. The relaxation method converged in 116 iterations where six restarts were required during the search.

The optimal thrust transient lunar transfer for a fixed flight time of 5.25 days is shown in Figure 5.2 for the mission profile specified. A close-up of the posigrade Lunar capture is given in Figure 5.3 A total of six engine mass flow switches are performed with two during the Earth departure and four during the Lunar capture. The boundary conditions on the switching function are different from those observed in Chapter 4 where both the initial and final points on the switching functions terminated at negative values indicating the initial and final thrust phases. Since the reactor starts at equilibrium at a minimal mass flow, the mass flow switching function must start at a positive value and only turn negative when the reactor has attained its rated power level and thrusting begins. The final mass flow switching value must be positive because of the necessary "cool-down" the reactor requires to return to the steady-state required by the boundary conditions. The total mass flow switching function looks like a parabola with zero initial and final points and does not convey much information. Close-ups of both the departure

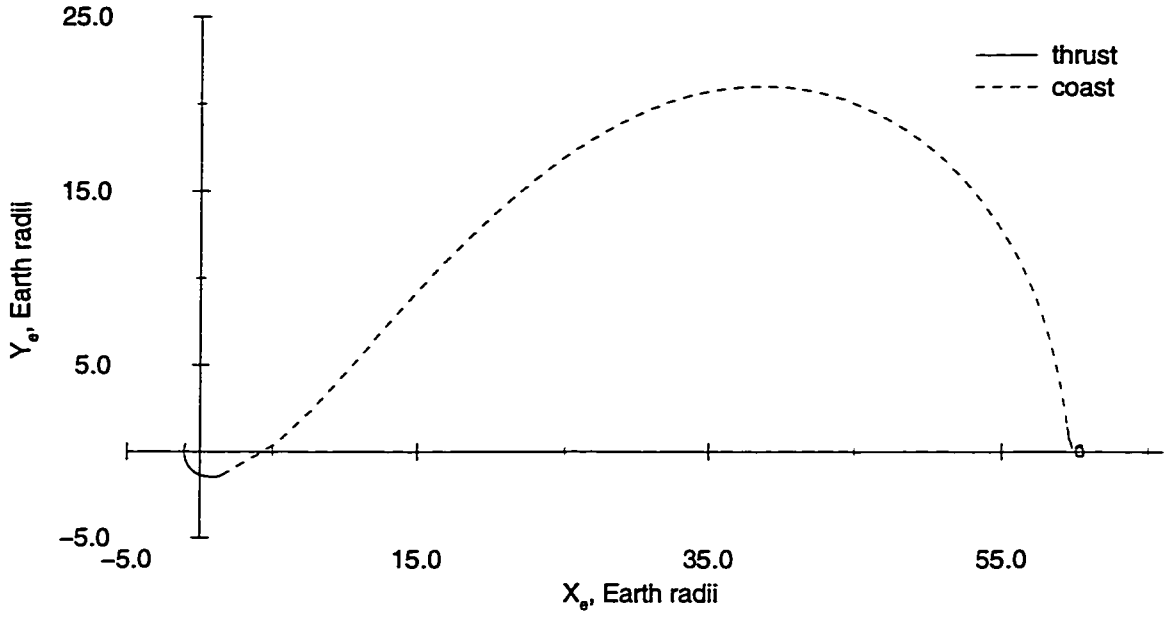

Figure 5.2 Optimal Transient NTR Lunar Transfer

and capture will provide a better understanding of how the switching function behaves. Figure 5.4 shows the mass flow switching function for the Earth departure. The first switching time occurs at just 12.2 seconds when the reactor is at full power and the thrusting begins. The thrust phase lasts for 50.757 minutes and the translunar coast begins.

The majority of the total transfer time is spent in the translunar coast which lasts for 5.087 days after which the Lunar capture begins. Figure 5.5 shows the mass flow switching function for the Lunar capture. The first thrust phase lasts 8.27 minutes and is followed by coast which has a duration of 2.5131 hours. The last thrust phase begins and lasts only 2.2965 minutes when the spacecraft enters the desired LLO. The last coast phase of duration 22.3 minutes is included to allow the reactor to shutdown and achieve steady-state conditions. The final mass in LLO is 222,846 kg for a performance mass

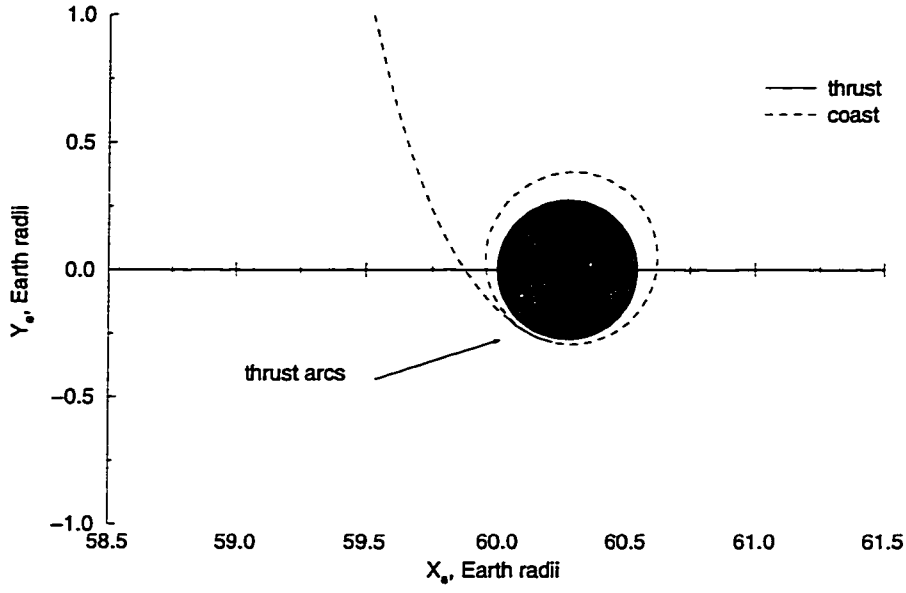

Figure 5.3 Optimal Transient NTR Lunar Transfer - Lunar Closeup

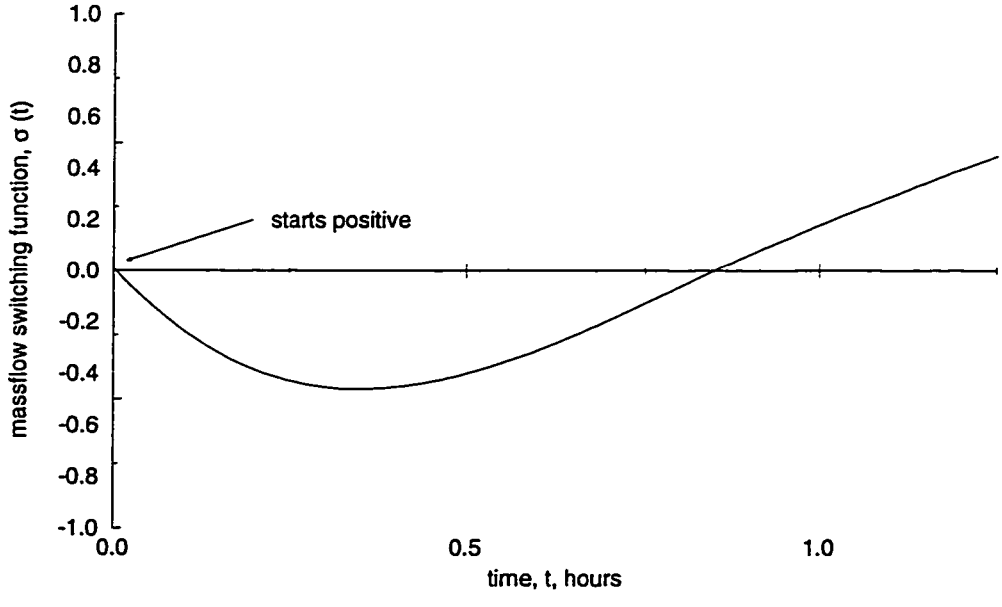

Figure 5.4 Transient NTR Lunar Transfer Mass flow Switching Function - Departure Closeup

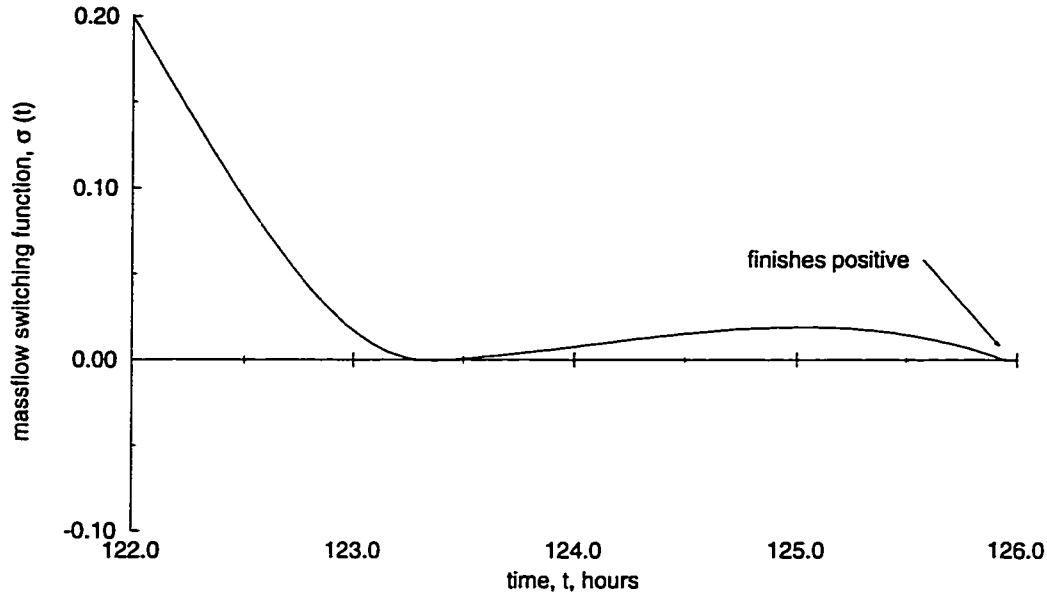

Figure 5.5 Transient NTR Lunar Transfer Mass flow Switching Function - Capture Closeup

fraction of 65.54 %.

The three thrust phases include time where the thrust is not a maximum and instead, the mass flow is used to cool the reactor. The cooling lowers the temperature of the core, thus lowering the chamber temperature of the propellant and thrust. The transient core conditions are shown for a startup, shutdown, and final capture burn phase in Figures 5.6, 5.7, and 5.8, respectively. The first two have already been discussed, while the final capture burn phase shows the core transient effects on a thrust phase "near" another thrust phase. The reactivity costate, which is the reactivity rate switching function, the normalized core temperature and normalized power are shown in Figure 5.6 for the Earth departure startup. The switching function follows the logical argument of having the highest possible reactivity rate $u_\rho^{\max}$ until the maximum reactivity is achieved which takes approximately 2.335 seconds. The reactivity rate is then set to zero until the core is near maximum power where the minimum reactivity rate is applied to put the

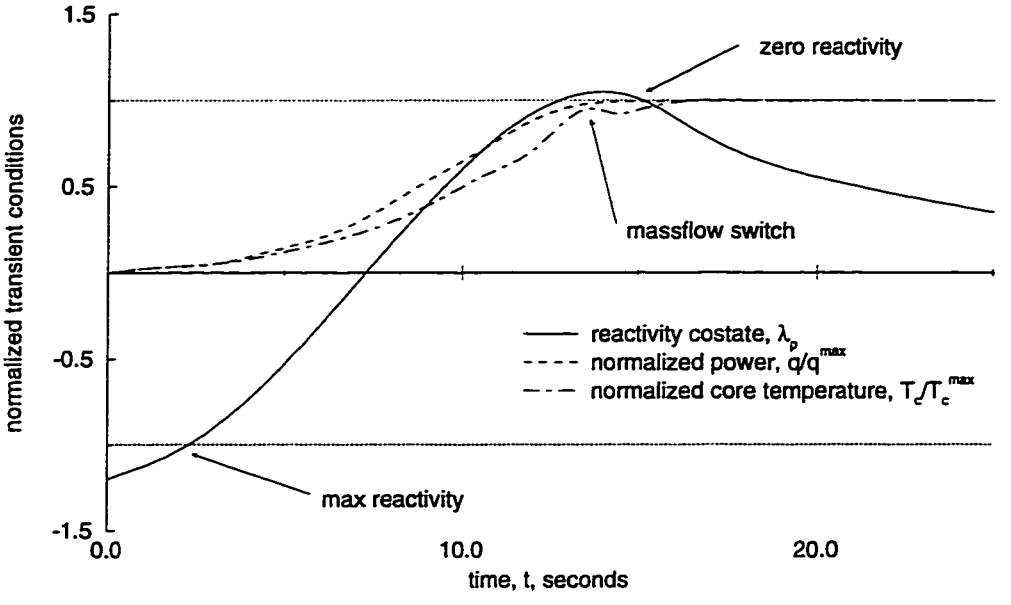

Figure 5.6 Startup Reactivity Rate Switching Function and Transient Conditions

core into a steady-state for the thrust phase. The entire startup process takes a little over 15.3 seconds where the mass flow is turned on 12.2 seconds into the startup phase. The effect of the increased mass flow is to lower the core temperature which lowers the propellant temperature exiting the core and thus the thrust. By keeping the reactivity on for the last 3.1 seconds the core is brought to its maximum allowable temperature to maximize the thrust. The transient conditions for the departure shutdown are shown in Figure 5.7. The control process for a shutdown shows the effects of the built up precursor power in the core by the long time required to lower both the core power and the core temperature. The reactivity rate is turned to it's lowest rate at 50.21 minutes and held until the minimum reactivity is achieved lowering the core power and temperature. The mass flow is switched off at 50.96 minutes causing a "spike" in the core temperature because the excess heat that was being removed by the propellant flow is now being absorbed by the reactor. This "spike" is the binding inequality constraint on

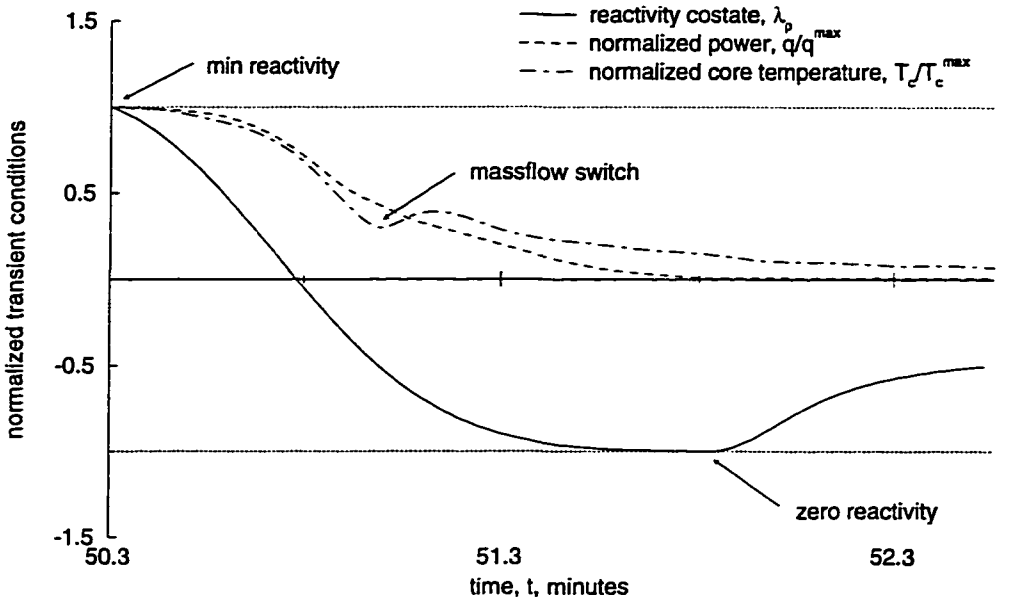

Figure 5.7 Shutdown Reactivity Rate Switching Function and Transient Conditions

the shutdown conditions because the rapid increase in core temperature can easily violate the temperature rate of change constraint. The mass flow is kept on during a shutdown phase until the core power is low enough to limit the "spike" to an acceptable level. The timing of the mass flow switch is then a trade-off between the extra propellant being used in decreasing thrust efficiency, since the temperature is also decreasing, versus the required temperature limitations on the core. The entire shutdown process takes slightly over 2.1 minutes.

The startup phase of the initial Lunar capture burn shows no effects of the previous departure thrust phase as would be expected because of the amount of time between the burns. However, the intermediate portion of the Lunar capture, i.e., the portion between the first and second capture thrust phases, poses a problem for the control strategy because of the small burn times (8.27 and 2.2965 minutes) and relatively short time between the burns (2.1 hours). The transfer solution solves this problem by altering

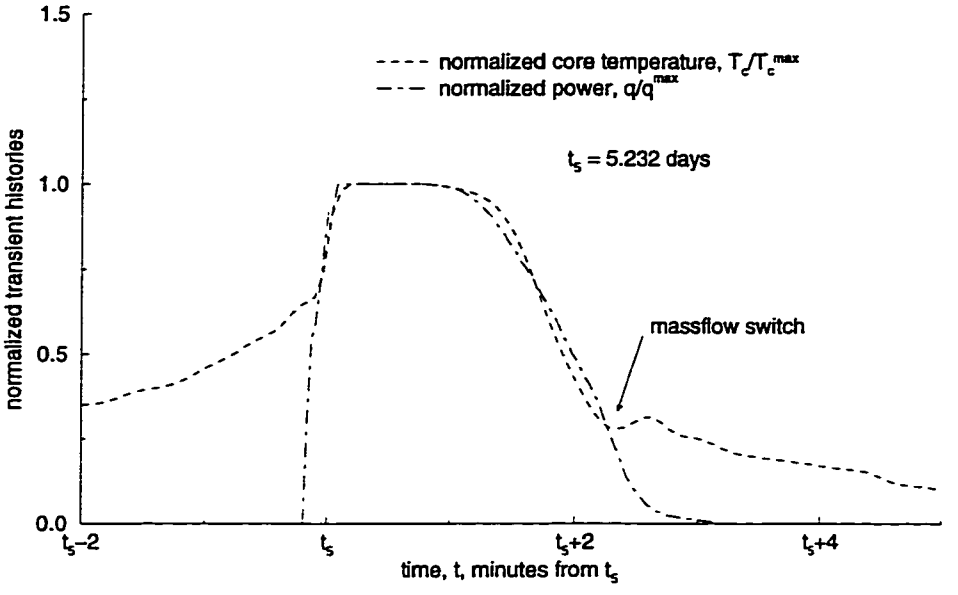

Figure 5.8 Final Capture Burn Transient Conditions

the shutdown and startup phases between the two capture burns. The shutdown phase of the first burn begins in a similar manner as the departure burn shutdown, but does not take the core to the minimal steady state, q^{min}. Instead, the core power is lowered to approximately 2.2 MW, and the propellant flow is still switched to a minimum. The temperature of the core rises over this period because the minimal mass flow can not remove all of the excess heat. The startup of the second capture thrust phase then begins with a high core temperature and requires less time for the core to achieve full power. The high core temperature improves the exhaust velocity of the startup transient phase, and the core can achieve full power in less than 3 seconds rather than the 10-15 normally required. The improved thrust values also allow the core to begin shutdown at an earlier time which allows the final coast in LLO to be as short as possible. Figure 5.8 shows the core temperature and power for the entire final thrust phase of the Lunar capture. The power curve is not greatly effected by the proximity of the other thrust phase while the temperature curve is very different. The thrust phase operates at full

power for only 1.12 minutes or 48.7 % of the entire phase while 51.2 % of the phase is spent bringing the temperature and core power down.

Comparison

The non-transient solution for the same flight time and mission profile was found and used as the initial estimate for the solution of the transient transfer. The non-transient transfer has a final mass of 239,771 kg for a mass performance ratio of 70.52% where The trajectory flight path is very similar to Figure 3.14.

A quick comparison between the solution presented here and previous work both verifies the results and shows the improved modeling effects. Two main studies have been done dealing with the effects of thrust transients with one being done by Zimmerman [37] and the other by Rivas [38]. Zimmerman used simple mass penalties to gauge the effect of thrust transients. The mass penalties were derived from experimental studies of the NERVA/KIWI programs and computer code simulations of the nuclear cores using the full neutron flux equations. The equations derived are

$$m_{startup} = 1.0142\left(10^{-3}\right) T_{nom} \tag{5.76}$$

$$m_{esc} = 0.0769\left(m\left(t_{departure}\right) - m\left(t_o\right)\right) \tag{5.77}$$

$$m_{cap} = 0.0773\left(m\left(t_f\right) - m\left(t_{capture}\right)\right) \tag{5.78}$$

$$m_{penalty} = m_{startup} + m_{esc} + m_{cap} \tag{5.79}$$

where $m_{startup}$ is the mass penalty for the startup of the NTR in kilograms, T_{nom} is the nominal thrust of the NTR in Newtons, m_{esc} is the mass penalty for the cool down after the departure thrust phase, $m\left(t_{departure}\right)$ is the mass at the end of the departure for the non-transient solution, m_{cap} is the penalty for the Lunar capture, $m\left(t_{capture}\right)$ is the mass at the beginning of the Lunar capture for the non-transient solution, and $m_{penalty}$ is the total mass penalty for a transient transfer. The study by Rivas used two different methods to approximate the transient effects. The first approximated the

Table 5.2 Transient Studies Comparison

case	final mass, kg	percent change
non-transient	239,771	-
transient	222,846	-7.06
Zimmerman	214,543	-10.52
Rivas	220,037	-8.23
	219,307	-8.53

experimental data from the NERVA/KIWI program with a set of analytical equations, and the second used curve-fits of the actual experimental data. Table 5.2 shows the separate solutions for the non-transient case, the transient case presented above, the results from applying the Zimmerman mass penalty method, and the results from applying the two methods of Rivas. The magnitudes of the extra propellant required are similar and the transient solution above is verified by having the solution comparable to the other models. The transient solution presented above yields a significantly better solution than the other models. Part of the improvement of the transient modeling over the models by Zimmerman and Rivas is the extra propellant the latter includes for structural cooling of the nozzle and fuel used in a bleed for powering a turbine, both of which account for 0.5-1% of the difference between the models. However, the transient model still improves the solution from 6-8% and includes a more detailed and accurate model of the nuclear core.

CHAPTER 6 CONCLUSIONS

The solution of minimum-fuel Lunar transfers is investigated in this thesis for a variety of mission definitions. The development of new numerical techniques is combined with theoretical insights and problem characteristics to both allow and ease the solution of these trajectories. The solutions are developed in the restricted circular three-body model of the Earth-Moon system and use either low-thrust electrical engines or nuclear thermal rockets (NTR) to provide propulsion. Most of the trajectories spiral about both the Earth and Moon and cause numerical difficulties because of this oscillatory nature. The sensitivity of the three-body model is compensated by using the model centered at both the Earth and Moon where the system version is determined by the problem formulation. An object-orientated class library was developed and used for all of the system models and solutions. The library makes full use of multi-inheritance, operator-overloading, and polymorphism to both eliminate unnecessary memory storage and increase computational performance.

The first set of solutions assumes a fixed engine firing sequence of thrust-coast-thrust to transfer the spacecraft between the LEO and the LLO. The solution of the full two point boundary value problem (2PBVP) is difficult but may be solved by a hierarchy of subproblems: maximum-energy departures about the Earth and captures about the Moon, all-coasting sub-optimal trajectories linking the departures and captures, and finally the solution of the full Earth-Moon trajectory. The second subproblem, which "patches" together the maximum-energy spirals to the coast transfer, is solved by a dynamic boundary evaluation (DBE) method. The DBE method eliminates the need for

large tables and/or storage of spiral solutions by solving the necessary coast boundary conditions during runtime. This decreases the computational time necessary to find a solution and is nearly autonomous, i.e., the user does not need to monitor the optimization. The complete optimal transfer is solved using a "hybrid" direct/indirect method. The thrust steering angle is parameterized by the costate equations which provides a very accurate approximation. A range of transfers were solved for varying initial thrust-to-weight ratios. Both posigrade and retrograde trajectories were found and the performance was compared among the solutions. The full 2PBVP is formulated and the solution is possible using the "hybrid" solution.

The optimal engine firing sequence for a power limited spacecraft is governed by the Pontryagin Minimum Principle and is investigated in Chapter 4. The solution of the full switching problem in the Earth-Moon three-body system is extremely difficult and a new problem definition is created and mixed-integer nonlinear programming (MINLP) is introduced. The problem is redefined along a discretization/collocation formulation which allows the MINLP algorithm to approximate the discrete switching structure. A numerical technique called fanning is created to keep the number of integer variables as low as possible for a given transfer solution. Dynamic mesh allocation is also used to both increase the accuracy of the discrete simulation and more accurately place the estimates of the engine switching times. A methodology of obtaining the solution of the 2PBVP for the full switching problem from the approximate solution returned by the MINLP approximation is also described and uses a combination of system characteristics and derivations to implement. A strategy of solving a general switching transfer from the MINLP approximation to the solution of the 2PBVP is also presented. Solutions for a range of flight-times and their switching structures are presented.

The effects of transients resulting from the use of NTR engines were modeled for optimal Lunar transfers. The nuclear core was simulated using the mono-energetic heat equations and the heat transfer by a lumped heat-exchange system. The combined

trajectory-propulsion system optimal problem is defined and solved for a specific mission profile. The solution found consisted of three thrusting and two coast phases where two thrust phases are used for the Lunar capture. The trajectory solution and optimal control histories for the transfer are presented for a startup, shutdown, and "altered" phase. The altered phase is the result of the last thrust phase's close time proximity to the first Lunar capture burn. This proximity alters how the core is controlled and is detailed in results section of Chapter 5. A comparison is given between this combined system and previous work on estimating the effects of the transients.

Future Research

Several areas of this dissertation can be extended for future research. The development of the DBE method introduces the possibility of combining the design of Lunar missions with the trajectory optimization process. The combination can take into account the "parking" orbits about the Earth and Moon and the energy requirements of launch vehicles to place the spacecraft in these orbits, the investigation of several different propulsion systems over a range of thrust values, and even include "out-and-back" mission scenarios for a Lunar transportation system. The switching problem of Chapter 4 is presented in a 2PBVP format which in general can not be solved using the relaxation method of Chapter 2 because of the discontinuous nature of the switching function and the inability to eliminate the mass flow control from the coupled equations. The development of the switching function and, more importantly, its derivative allows a different formulation of the 2PBVP. The mass costate equation is uncoupled from the rest of the state/costate systems and the switching function derivative equation can be substituted for the mass costate equation. This substitution of σ for λ_m can be used to eliminate the the control in the rest of the state/costate system and the relaxation method can then be applied to the 2PBVP. Difficulties arise in the calculation of the

Jacobian of this new 2PBVP, which turns out to be singular, which would need to be overcome for the method to be viable. The NTR combined trajectory/propulsion system of Chapter 5 simulates the nuclear core and heat transfer model to model the transients resulting from the NTR use. Another approach for modeling the transients would be to replace the heat-exchange dynamics with a system that requires a certain propellant mass flow for a given power level. This approach is very similar to that used by Sachs and Dinkelmann [11] and allows a finer modeling of the propellant flow needs to actual experimental data.

APPENDIX A GENERALIZED CROSS DECOMPOSITION (GCD)

The Generalized Cross Decomposition, GCD, consists of two phases: the primal and dual subproblem phase (Phase I) and the master problem phase (Phase II). In Phase I, the primal subproblem provides an upper bound on the sought solution and the Lagrange multipliers for the dual subproblem. The dual subproblem provides a lower bound on the solution and supplies a current integral vector for the primal subproblem. Both the primal and dual subproblems provide cuts for the master problem in Phase II. An iteration of the GCD algorithm solves a primal and dual subproblem and a primal convergence test is applied to the current integral vector, while a dual convergence test is applied to the dual Lagrange multipliers. If any convergence test fails, Phase II is executed which solves a master problem and then returns to Phase I. The key idea is to make extensive use of Phase I and limit as much as possible the use of Phase II by the application of appropriate convergence tests. This assumes that the solution of the master problem in Phase II is much more difficult and CPU time consuming then the solution of the subproblems in Phase I which is nearly always the case. The subproblems and master problems of both phases are determined below, along with a description and discussion of the chosen convergence tests. Finally, the general algorithm is given which was implemented for this work. The complete derivation is not given, nor the necessary background in duality theory since this is beyond the scope of this work. The details of the derivation and other aspects of the algorithm can be found in Holmberg [19], Van

Roy [40], Floudas [20], Benders [41], and Geoffrion [42].

The general problem statement which will be considered follows:

$$\begin{aligned} \min_{x,y} \quad & f(x,y) \\ \text{subject to:} \quad & h(x,y) = 0 \\ & g(x,y) \leq 0 \end{aligned} \tag{A.1}$$

with $x \in \Re^n$, $y \in Y = \{0,1\}^q$, $h \in \Re^{m_e}$, and $g \in \Re^{m_i}$. There are two possible cases resulting from trying to solve Equation (A.1): feasible and infeasible solutions. An infeasible solution is one where the constraints can not be satisfied. Two Lagrange functions are then defined, one for a feasible solution

$$L(x,y,\lambda,\mu) = f(x,y) + \lambda^T h(x,y) + \mu^T g(x,y) \tag{A.2}$$

and one for an infeasible solution

$$\bar{L}\left(x,y,\bar{\lambda},\bar{\mu}\right) = \bar{\lambda}^T h(x,y) + \bar{\mu}^T g(x,y) \tag{A.3}$$

Development of Problems

Phase I - Subproblems

The primal subproblem results from fixing the y-vector to a particular $0-1$ combination denoted as y^k. The problem can be stated as

$$\begin{aligned} \min_{x} \quad & f\left(x,y^k\right) \\ \text{subject to:} \quad & h\left(x,y^k\right) = 0 \\ & g\left(x,y^k\right) \leq 0 \end{aligned} \tag{A.4}$$

A feasible and infeasible solution to Equation (A.4) is possible. A feasible solution yields a corresponding float vector, x^k, the optimal objective function value, $P^k = P\left(y^k\right)$, and

the Lagrange multipliers λ^k and μ^k. The dual subproblem uses λ^k and μ^k and takes the form

$$\begin{aligned} \min_{x,y} \quad & f(x,y) + \left(\mu^k\right)^T g(x,y) \\ \text{subject to:} \quad & h(x,y) = 0 \end{aligned} \tag{A.5}$$

When an infeasible solution is detected, a feasibility problem is formulated and takes the form:

$$\begin{aligned} \min_{x} \quad & \|\alpha\| \\ \text{subject to:} \quad & h(x,y) = 0 \\ & g(x,y) \leq \alpha \\ & \alpha \geq 0 \end{aligned} \tag{A.6}$$

where α is a vector of length m_i which is restricted to be always positive. Note that $\|\alpha\| = 0$ is a feasible point. The solution of Equation (A.6) provides the Lagrange multipliers $\bar{\lambda}^l$ and $\bar{\mu}^l$ and the Lagrange value, $\bar{L}^l = \bar{L}\left(x, y, \bar{\lambda}^l, \bar{\mu}^l\right)$. The dual subproblem then takes the form

$$\begin{aligned} \min_{x,y} \quad & \left(\mu^l\right)^T g(x,y) \\ \text{subject to:} \quad & h(x,y) = 0 \end{aligned} \tag{A.7}$$

Note that the solution of Equation (A.7) does not provide any bound on Equation (A.1), and it can only provide a dual cut that will eliminate $\bar{\mu}^l$ from further consideration.

Phase II - Master Problems

The development of Phase II is identical to the generalized benders decomposition of Benders [41] and will not be repeated here. The final form of the relaxed primal master problem is

$$\min_{y,\mu_c} \quad \mu_c \tag{A.8}$$

$$\begin{aligned} \text{subject to:} \qquad \mu_c &\geq L\left(x^k, y, \lambda^k, \mu^k\right) k = 1, 2, \ldots, K \\ 0 &\geq \bar{L}\left(x^l, y, \lambda^l, \mu^l\right), l = 1, 2, \ldots, L \end{aligned}$$

where K is the number of found feasible cuts and L is the number of found infeasible cuts. Note that this formulation uses the assumption of Geoffrion [42] that a local linearization about the previously found feasible and infeasible points will provide valid cuts. The primary significance is that the method can no longer claim global optimality, but a local minimum is still assured.

The derivation of the Lagrange relaxation master problem employs Lagrangian duality and considers the dualization of the $g(x, y)$ inequality constraints. The dual takes the following form:

$$\max_{\mu \geq 0} \left[\begin{array}{l} \min_{x,y} f(x, y) + \mu^T g(x, y) \\ \qquad \text{subject to:} \quad h(x, y) \end{array} \right] \tag{A.9}$$

The inner problem of Equation (A.9) is parametric in μ, and for a fixed value of μ it corresponds to the dual subproblem of Equation (A.5) in Phase I. Denoting the solution of the dual as $\left(x^k, y^k\right)$, assuming feasibility, and define

$$h^k(\mu) = f\left(x^k, y^k\right) + \mu^T g\left(x^k, y^k\right) \tag{A.10}$$

then the relaxed Lagrange relaxation master problem becomes

$$\begin{aligned} \max_{\mu, \acute{\mu}_c} \quad & \acute{\mu}_c \\ \text{subject to:} \quad & \acute{\mu}_c \leq h^k(\mu), k = 1, 2, \ldots, K \\ & \mu \geq 0 \end{aligned} \tag{A.11}$$

The solution of Equation (A.11) provides a valid upper bound on the primal problem only if that problem is convex, therefore it can only be used as a heuristic. In the case of infeasibility, the following cuts are introduced

$$h^l(\bar{\mu}) = \bar{\mu}^T g\left(x^k, y^k\right) \tag{A.12}$$

Convergence Tests

The convergence tests are used to provide

1. upper bound improvement
2. lower bound improvement
3. cut improvement

An upper bound improvement corresponds to a decrease in the upper bound, UBD, obtained by the primal subproblem, Equation (A.4). A lower bound improvement corresponds to an increase in the lower bound, LBD, obtained by the dual subproblem, Equation (A.5). A cut improvement corresponds to generating a new cut which becomes active and is not dominated by the cuts generated in previous iterations. Two cuts can be found from Equation (A.8), primal cut improvement, and from Equation (A.11), a Lagrange relaxation cut improvement.

Three tests are required and can be described as

CTP This test determines if a current y-vector, denoted here by y^c, can provide an upper bound improvement. If

$$L\left(x^k, y^c, \lambda^k, \mu^k\right) < UBD \qquad \text{for} \qquad k = 1, 2, \ldots, K$$
$$\bar{L}\left(x^l, y^c, \lambda^l, \mu^l\right) \leq 0 \qquad \text{for} \qquad l = 1, 2, \ldots, L$$

then y^c does provide an upper bound improvement and the test is passed.

CTD This test determines if a y^c can provide a lower bound improvement. If

$$h^k(\mu^c) > LBD \qquad \text{for} \qquad k = 1, 2, \ldots, K \tag{A.13}$$

where μ^c corresponds to the multipliers for y^c, then the test is passed.

CTDU This test determines if a relaxation cut improvement is valid. If

$$h^l(\bar{\mu}^c) > 0 \qquad \text{for} \qquad l = 1, 2, \ldots, L \tag{A.14}$$

where $\bar{\mu}^c$ corresponds to the multipliers for y^c, then the test is passed.

The first condition of the CTP test and the CTD test are "value convergence" tests because they rely on feasible problems, while the second condition of the CTP test and the CTDU test are "feasibility convergence" tests because they rely on feasibility problems for infeasible solutions.

The proof of a convergence in a finite number of iterations is given by Holmberg [19] and relies completely on the CTP test. This does not mean however that the CTD test cannot provide convergence, only that it can not be assured of doing so in a finite number of steps.

The algorithm in the next section also makes use of a simple bounds test of the form

$$\text{If} \qquad |\, UBD - LBD \,| < \epsilon, \qquad \text{stop} \tag{A.15}$$

This test will be called the BT test and provides the stopping mechanism in the algorithm.

Algorithm

A list of definitions will allow a more elegant representation of the GCD algorithm:

- The solution of Equation (A.4) will be denoted as $P\left(y^k\right)$.
- The solution of Equation (A.11) will be denoted as $RLRM$.
- The solution of Equation (A.5) will be denoted as $D\left(\mu^k\right)$.
- The solution of Equation (A.6) will be denoted as $FP\left(y^k\right)$.

- The solution of Equation (A.7) will be denoted as $FD\left(\mu^l\right)$.
- The solution of Equation (A.8) will be denoted as RPM.

This list and the convergence tests are used in the algorithm which follows:

1. Given the initial estimate y^1 and the convergence tolerance ϵ. Set the upper bound to infinity, $UBD = \infty$, the lower bound to minus infinity, $LBD = -\infty$, and initialize the feasible solution counter, $k = 1$, and the infeasible solution counter, $l = 1$.

2. Attempt to solve $P\left(y^k\right)$. If the problem is feasible go to 3, otherwise the problem is infeasible and go to 4.

3. "FEASIBLE" Let $\hat{x}$ be the solution to the feasible problem with $\hat{\lambda}$ and $\hat{\mu}$ the Lagrange multipliers. Set the upper bound to

$$UBD = \min_k P\left(y^k\right) \tag{A.16}$$

and check the bounds, i.e. apply the BT test. If the test is passed stop.

   (a) Apply the CTD test. If passed go to 3c, otherwise go to 3b.

   (b) Solve the relaxed Lagrangian master problem, $RLRM$, which provides a new UBD and μ.

   (c) Solve the dual subproblem $D(\mu)$ and update the lower bound,

$$LBD = \max\left(LBD, \hat{D}\right) \tag{A.17}$$

   (d) Apply a bounds test, BT. If it is passed stop, otherwise go to 5.

4. "INFEASIBLE" Solve the feasibility problem $FP\left(y^k\right)$ and determine the feasible Lagrange multipliers, $\bar{\lambda}$ and $\bar{\mu}$.

(a) Apply the CTDU test. If passed go to 4b, otherwise to to 4c.

(b) Solve the dual feasibility problem $FD\left(\bar{\lambda}\right)$, and denote the solution as $(\hat{x}, \hat{y})$ and increment feasibility counter, $l = l + 1$, and $y^l = \hat{y}$. Go to 5.

(c) Solve the relaxed primal master problem, RPM, to find $(\hat{y}, \hat{\mu}^c)$. Update the lower bound:

$$LBD = \hat{\mu}^c \tag{A.18}$$

Apply the BT test. If passed stop, otherwise increment $k = k+1$ and $y^k = \hat{y}$. Go to 2.

5. Apply the CTP test. If passed go to 3, otherwise go to 4c.

Figure A.1 gives a flow chart of this algorithm. Note that problems D (Equation (A.5)), FD (Equation (A.7)), and RPM (Equation (A.8)) require the solution of separate MINLP problems. These problems are solved using a Branch and Bound algorithm proposed by Borchers and Mitchell [18] which is described in Chapter 2.

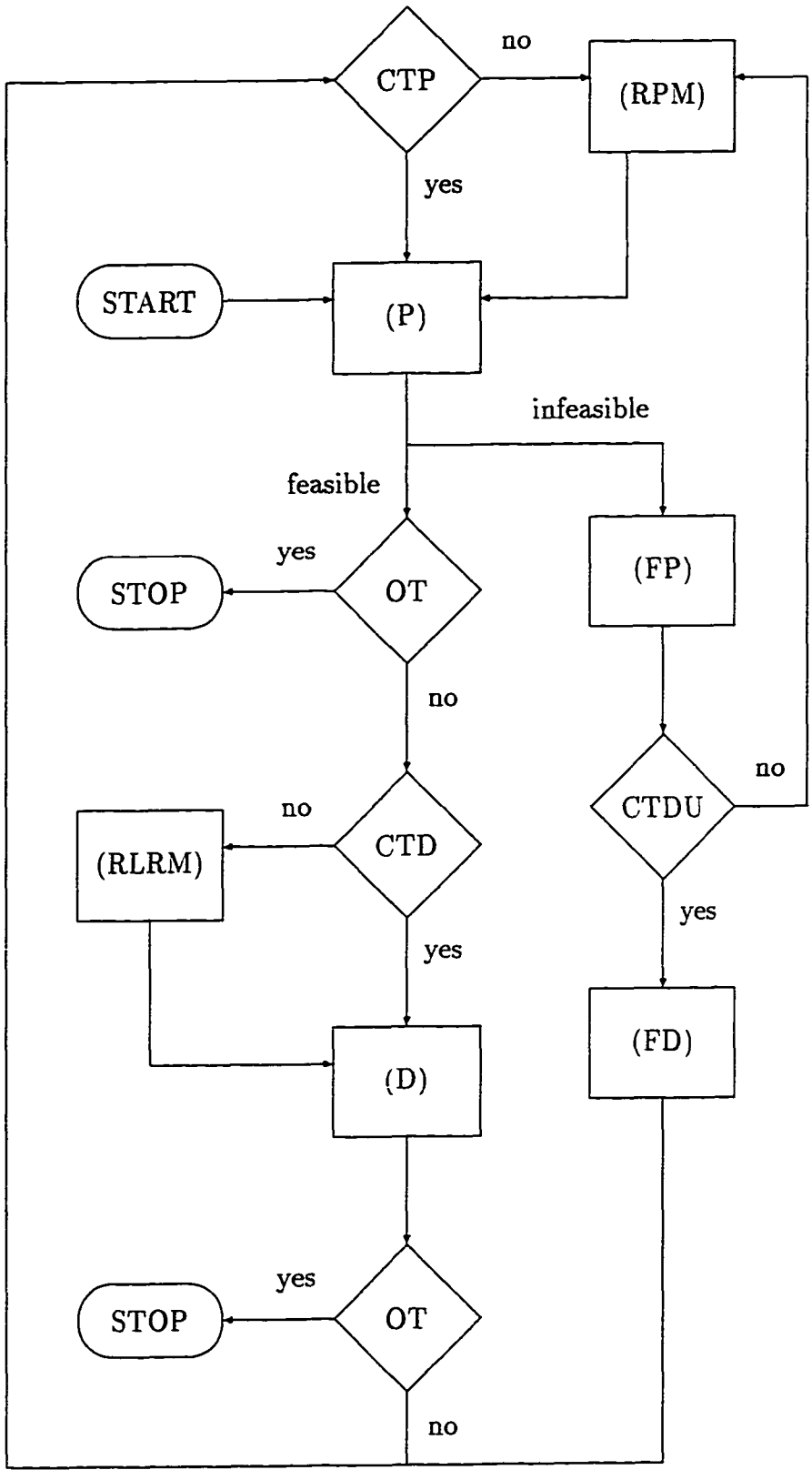

Figure A.1 Flow Diagram for GCD Algorithm

APPENDIX B CLASS LIBRARY HEADER FILES

The following code listings are C++ headers files which describe the object library structure described in Chapter 2. The code is commented throughout by standard C, the backslash and asterisks, and C++, the double backslash, commenting methods.

```
/*
    Class Dynamics

    Base class of dynamic equations. Member variables are:

      nVariables := number of states in dynamics
      nControls  := number of controls in dyanmics

    Member functions are:

      dyn(t,x,u,dxdt) := given the independent variable t,
         the current state x and current control u, return
         the right-hand sides in dxdt. Function declared
         pure virtual.
      jacobian(t,x,u,dfdx) := given the independent variable
         t, the current state x and control u, return the
         jacobian of the right-hand sides in dfdx. Default
         is a numerical finite-difference formula.
*/
class Dynamics {
  protected:
    short nVariables;
    short nControls;
  public:
    Dynamics(short n,short m);
    ~Dynamics();
    short getSystemSize() {return (nVariables);};
```

```
    short getControlSize() {return (nControls);};
    virtual void dyn(double t,double x[],double u[],double dx[]) = 0;
    virtual void jacobian(double t, double x[], double u[], double *dfdx[]);
};

/*
    Class TwoBody

    Implements two-body Newtonian dynamics with a thrusting spacecraft.
    Five states are defined as follows:

       r[0] := radius
       r[1] := polar angle
       r[2] := radial velocity
       r[3] := tangential velocity
       r[4] := mass of spacecraft

    Two control's are possible:

       u[0] := thrust steering angle
       u[1] := massflow

    There are no bounds on the thrust steering angle, u[0], and an
    upper limit on massflow, u[1].
*/
class TwoBody : public Dynamics , public Spacecraft {
  protected:
    double mu;
    double gravityParameter() {return (mu);};

  public:
    TwoBody(double t2w,double isp,double mi);
    TwoBody(double gp,double t2w,double isp,double mi,double DU,double TU,double MU);
    ~TwoBody();
    void dyn(double t, double r[], double u[], double dr[]);
    void jacobian(double t, double r[], double u[], double *dfdr[]);
};

/*
  Class ThreeBody and ThreeBodyLunar
```

```
   Implements Newtonian dynamics in a restriced circular three-body
   system with a thrusting spacecraft.  Five states are defined as
   follows:

     r[0] := radius
     r[1] := polar angle
     r[2] := radial velocity
     r[3] := tangential velocity
     r[4] := mass of spacecraft

   Two controls are implemented:

     u[0] := thrust steering angle
     u[1] := massflow rate

   There are no bounds on the thrust steering angle, u[0], and the
   massflow rate is limited between zero and some upper limit, i.e.,
   0 <= u[1] <= max_limit.  The ThreeBodyLunar class implements the
   same system but centered on the secondary body.
*/
class ThreeBody : public Dynamics , public Spacecraft {
  protected:
    double mu_p; // primary mass parameter
    double mu_s; // secondary mass parameter
    double dist; // distance between two masses
    double omeg; // rotational velocity of moon
    // useful constants;
    double twoOmega;
    double omega2;
    double oneOverdist2;
  public:
    ThreeBody(double t2w,double isp,double mi);
    ThreeBody(double gp1, double gp2, double d, double w,double t2w,
              double isp,double mi,double DU,double TU,double MU);
    ~ThreeBody();
    void dyn(double t, double r[], double u[], double dr[]);
    void jacobian(double t, double r[], double u[], double *dfdr[]);
};
```

```
class ThreeBodyLunar : public Dynamics , public Spacecraft {
  protected:
    double mu_p; // primary mass parameter
    double mu_s; // secondary mass parameter
    double dist; // distance between two masses
    double omeg; // rotational velocity of
    // useful constants;
    double twoOmega;
    double omega2;
    double oneOverdist2;
  public:
    ThreeBodyLunar(double t2w,double isp,double mi);
    ThreeBodyLunar(double gp1,double gp2,double d,double w,double t2w,
                   double isp,double mi,double DU,double TU,double MU);
    ~ThreeBodyLunar();
    void dyn(double t, double r[], double u[], double dr[]);
    void jacobian(double t, double r[], double u[], double *dfdr[]);
};

/*
  Numerical Class which integrates a set of ODE. The ODE's are
  evaluated by the pure virtual member function

    odes(x,y,dydx) = 0;

  where given independent variable x and current dependent vector y,
  return the ODE's in dydx. The default integrator is a Bulirsch-Stoer
  extrapolation scheme, but a Runge-Kutte scheme is available as well
  as a semi-implicit stiff integration method.
*/
class Integration {
  private:
    double min(double a,double b) {return ((a) <= (b) ? (a) : (b));}
    double max(double a,double b) {return ((a) >= (b) ? (a) : (b));}
    double sign(double a,double b) {return ((b) >= 0.0 ? fabs(a) : -fabs(a));}
    void ludcmp(double **a,short *indx,double &d);
    void lubksb(double **a, short *indx, double b[]);
  protected:
    short nok;
    short nbad;
```

```
    short numODEs;
    double eps;
    double minstep;
    double **db;
    double *xb;
    double *yscal;
    virtual void odes(double x,double y[],double dydx[]) = 0;
    virtual void jacobn(double x,double y[],double dfdx[],double **dfdy);
    void mmid(double y[],double dydx[],double xs,double htot,short nstep,double yout[]);
    void bsstep(double y[],double dydx[],double *xx,double htry,double *hdid,double *hnext);
    void pzextr(short iest,double xest,double yest[],double yz[],double dy[]);
    void rkqs(double y[],double dydx[],double *x,double htry,double *hdid,double *hnext);
    void rkck(double y[],double dydx[],double x,double h,double yout[],double yerr[]);
    void stifbs(double y[],double dydx[],double *xx,double htry,double *hdid,double *hnext);
    void simpr(double y[],double dydx[],double dfdx[],double **dfdy,double xs,double htot,
               short nstep,double yout[]);
  public:
    Integration(short n);
    Integration(short n,double accuracy,double hmin);
    ~Integration();
    virtual void Integrate(double xo,double xf,double yo[],short n,double acc,double hmin);
    virtual void Integrate(double xo,double xf,double yo[]);
    virtual void Integrate(double xo,double xf,double yo[],long kmax,long *kount,double dxsav,
                           double *xp,double *yp[]);
    virtual void Integrate_rk(double xo,double xf,double yo[],short n,double acc,double hmin);
    virtual void Integrate_rk(double xo,double xf,double yo[]);
    virtual void Integrate_rk(double xo,double xf,double yo[],long kmax,long *kount,double dxsav,
                           double *xp,double *yp[]);
    virtual void Integrate_stiff(double xo,double xf,double yo[],short n,double acc,double hmin);
    virtual void Integrate_stiff(double xo,double xf,double yo[]);
    virtual void Integrate_stiff(double xo,double xf,double yo[],long kmax,long *kount,double dxsav,
                           double *xp,double *yp[]);
};

/*
  General Optimal Control Problem

    min J = phi[r(tf),tf] + integral[ L(r,u,t)] dt

  subject to dr/dt = f(r, u, t)
```

```
  and satisfying: Psi[r(to), r(tf)] = 0

  Hamiltonian:
    H = L + l'*f

  state equations:
    dr/dt = f

    costate equations:
dl/dt = -(dL/dx)' - (df/dx)'l

    stationarity condition:
dH/du = 0

note: a time-of-flight control parameter is allowed
     for the integration, i.e.,

         dr/dtau = alpha * f(r, u, alpha*p)

     where 0 <= tau <= 1.0. This allows a fixed-time
     of flight, tau now goes between zero and one,
     and a variable total time-of-flight, alpha. The
     parameter is entered when the integration is
     performed by the object.
*/
class OptimalControl : public Integration {
  protected:
    Dynamics *model;
    double alpha;
  public:
    OptimalControl(Dynamics *model);
    ~OptimalControl();
    void TimeOfFlight(double tof);
    double Hamiltonian(double t, double r[], double u[], double l[]);
    virtual void odes(double t, double x[], double dx[]);
    virtual void control(double t, double state[],double costate[], double u[]) = 0;
    virtual double phi(double t, double x[]);
    virtual void dPhidx(double t, double x[], double dphi[]);
    virtual double L(double t, double x[], double u[]);
```

```
    virtual void dLdx(double t, double x[], double u[], double dL[]);
};

/*
  Base class which implements common spacecraft behavior, e.g.,
  given exhaust velocity and massflow determine thrust. Primary
  cause of creation is to allow transient conditions in rocket
  operation, i.e., if exhaust velocity is lowered what happens
  to Isp, thrust, etc.
*/
class Spacecraft {
  protected:
    double scExhaustVelocity; // exhaust velocity
    double scMassflow;        // mass flow
    double scInitialMass;     // initial mass
  public:
    Spacecraft();
    Spacecraft(double thrust2weight,double isp,double initialmass,
               double DU = 1.0,double TU = 1.0,double MU = 1.0);
    ~Spacecraft();
    virtual double ExVelocity() {return (scExhaustVelocity);};
    virtual double Massflow() {return (scMassflow);};
    virtual double Thrust() {return (ExVelocity()*Massflow());};
    virtual double InitialMass() {return (scInitialMass);};
};

/*
  Matrix Classes - standard matrix implementation
*/
class Matrix {
  protected:
    short type;
    short rows;
    short cols;
    double **coeff;
  public:
    Matrix();
    Matrix(short n,short m);
    ~Matrix();
    virtual double *operator [] (short r) {return coeff[r];};
```

```
    virtual double &operator () (short i,short j) const;
    virtual void convertToBaseOne();
    virtual void convertToBaseZero();
    short getType() {return (type);};
    short getRows() {return (rows);};
    short getCols() {return (cols);};
    virtual void   setValue(double x,short i,short j) {coeff[i][j] = x;};
    virtual double getValue(short i,short j) const {return (coeff[i][j]);};
    virtual void   getValue(Matrix *A) const;
    friend ostream &operator << (ostream &os,Matrix &mat);
    virtual Matrix &operator = (const Matrix &rval);
    virtual void scalarMultiply(double a);
    friend void Sum(const Matrix &lval,const Matrix &rval,Matrix *result);
};

// symmetric matrix class
class SymMatrix : public Matrix {
  public:
    SymMatrix(short n);
    ~SymMatrix();
    virtual double &operator () (short i,short j) const;
    virtual void   setValue(double x,short i,short j);
    virtual double getValue(short i,short j) const;
    virtual void   getValue(Matrix *A) const;
    SymMatrix &operator = (const SymMatrix &rval);
    virtual void scalarMultiply(double a);
    friend void Sum(const SymMatrix &lval,const SymMatrix &rval,SymMatrix *result);
};

// banded block matrix class
class BandMatrix : public Matrix {
  static double zero;
  protected:
    char  off;
    short row_band;
    short col_band;
    short num_band;
    short offset;
  public:
    BandMatrix(short n,short m,short n_band,short m_band);
```

```
    ~BandMatrix();
    virtual double &operator () (short i,short j) const;
    virtual void convertToBaseOne() {off = 1;};
    virtual void convertToBaseZero() {off = 0;};
    short getRowBand() {return (row_band);};
    short getColBand() {return (col_band);};
    virtual void setValue(double x,short i,short j);
    virtual double getValue(short i,short j) const;
    virtual void   getValue(Matrix *A) const;
    BandMatrix &operator = (const BandMatrix &rval);
    virtual void scalarMultiply(double a);
    friend void Sum(const BandMatrix &lval,const BandMatrix &rval,BandMatrix *result);
};

// trapezoid matrix class
class TrapMatrix : public Matrix {
  static double zero;
  protected:
    char  off;
    short numV;
    short numVU;
    short numS;
    short offset;
    short no_psi;
    short nf_psi;
  public:
    TrapMatrix(short ns,short nv,short nu,short no,short nf);
    ~TrapMatrix();
    virtual double &operator () (short i,short j) const;
    virtual void convertToBaseOne() {off = 1;};
    virtual void convertToBaseZero() {off = 0;};
    virtual void   setValue(double x,short i,short j);
    virtual double getValue(short i,short j) const;
    virtual void   getValue(Matrix *A) const;
    TrapMatrix &operator =(const TrapMatrix &r);
    virtual void scalarMultiply(double a);
    friend void Sum(const TrapMatrix &lval,const TrapMatrix &rval,TrapMatrix *result);
};

// hermite-simpson matrix class
```

```
class HermSimpMatrix : public Matrix {
  static double zero;
  protected:
    char  off;
    short numV;
    short numVU;
    short numVUU;
    short num2VUU;
    short numS;
    short no_psi;
    short nf_psi;
  public:
    HermSimpMatrix(short ns,short nv,short nu,short no,short nf);
    ~HermSimpMatrix();
    virtual double &operator () (short i,short j) const;
    virtual void convertToBaseOne() {off = 1;};
    virtual void convertToBaseZero() {off = 0;};
    virtual void   setValue(double x,short i,short j);
    virtual double getValue(short i,short j) const;
    virtual void   getValue(Matrix *A) const;
    HermSimpMatrix &operator =(const HermSimpMatrix &r);
    friend void Sum(const HermSimpMatrix &lval,const HermSimpMatrix &rval,HermSimpMatrix *result);
};
```

APPENDIX C ADDITIONAL NECESSARY CONDITIONS

The necessary conditions for an optimal control problem with initial and terminal state constraints and control parameters is derived in this section. The general problem statement is:

Find $u(t)$, $t_o \leq t \leq t_f$, and the control parameter vector $\alpha \in \Re^p$ which minimizes

$$J = \phi[x(t_f), \alpha, t_f] + \int_{t_o}^{t_f} L(x(t), u(t), \alpha, t)\, dt \tag{C.1}$$

subject to

$$\dot{x} = f(x(t), u(t), \alpha, t) \tag{C.2}$$

with the general initial and terminal state constraints:

$$\psi[x(t_o), x(t_f)] = 0 \tag{C.3}$$

where $x \in \Re^n$ and $\psi \in \Re^q$. An augmented performance index, J', is define as

$$J' = \phi[x(t_f), \alpha, t_f] + \nu^T \psi[x(t_o), x(t_f)] + \int_{t_o}^{t_f} \left[L + \lambda^T (f - \dot{x})\right] dt \tag{C.4}$$

where ν is a q-vector of constant multipliers and λ is a n-vector of costate multipliers. The Hamiltonian is introduced for convenience and is defined as

$$H = L + \lambda^T f \tag{C.5}$$

The augmented performance index can then be written as

$$J' = \phi[x(t_f), \alpha, t_f] + \nu^T \psi[x(t_o), x(t_f)] + \int_{t_o}^{t_f} H - \lambda^T \dot{x} dt \tag{C.6}$$

The last term can be integrated by parts to yield

$$\begin{aligned} J' &= \phi\left[x\left(t_f\right), \alpha, t_f\right] + \nu^T \psi\left[x\left(t_o\right), x\left(t_f\right)\right] - \lambda^T\left(t_f\right) x\left(t_f\right) + \lambda^T\left(t_o\right) x\left(t_o\right) \quad \text{(C.7)} \\ &+ \int_{t_o}^{t_f} \left(H + \dot{\lambda}^T x\right) dt \end{aligned}$$

Now, expand J' to a first-order Taylor series about control variations in $u(t)$ and α:

$$\begin{aligned} \delta J' &= \left[\frac{\partial \phi}{\partial x} + \left(\frac{\partial \psi}{\partial x}\right)^T \nu - \lambda\right]_{t=t_f}^T \delta x\left(t_f\right) + \left[\frac{\partial \phi}{\partial t} + \left(\frac{\partial \psi}{\partial t}\right)^T \nu + H\right]_{t=t_f} \delta t_f \quad \text{(C.8)} \\ &+ \psi \delta \nu + \left[\left(\frac{\partial \psi}{\partial x}\right)^T \nu + \lambda^T\right]_{t=t_o} \delta x\left(t_o\right) + \frac{\partial \phi}{\partial \alpha} \delta \alpha \\ &+ \int_{t_o}^{t_f} \left[\left(\frac{\partial H}{\partial x} + \dot{\lambda}\right)^T \delta x \left(\frac{\partial H}{\partial u}\right)^T \delta u + \frac{\partial H}{\partial \alpha} \delta \alpha + \left(\frac{\partial H}{\partial \lambda} - \dot{x}\right)^T \delta \lambda\right] dt \end{aligned}$$

The first variation must be equal to zero for a minimum to occur and setting the independent increments to zero yields:

$$\dot{x} = \frac{\partial H}{\partial \lambda} = f\left(x, u, \alpha, t\right) \quad \text{(C.9)}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \quad \text{(C.10)}$$

$$0 = \frac{\partial H}{\partial u} \quad \text{(C.11)}$$

$$0 = \psi\left[x\left(t_o\right), x\left(t_f\right)\right] \quad \text{(C.12)}$$

$$0 = \frac{\partial \phi}{\partial \alpha} + \int_{t_o}^{t_f} \frac{\partial H}{\partial \alpha} dt \quad \text{(C.13)}$$

$$\begin{aligned} 0 &= \left[\left(\frac{\partial \psi}{\partial x}\right)^T \nu + \lambda\right]_{t=t_o} \delta x\left(t_o\right) + \left[\frac{\partial \phi}{\partial x} + \left(\frac{\partial \psi}{\partial x}\right)^T \nu - \lambda\right]_{t=t_F}^T \delta x\left(t_f\right) \quad \text{(C.14)} \\ &+ \left[\frac{\partial \phi}{\partial t} + \left(\frac{\partial \psi}{\partial t}\right)^T \nu + H\right]_{t=t_f} \delta t_f \end{aligned}$$

Equation (C.9) defines the state equations, Equation (C.10) defines the costate equations, and Equation (C.12) imposes the required initial and terminal state constraints. Equation (C.11) is the stationarity condition and Equation (C.13) is the new necessary condition when control parameters are present. Equation (C.14) defines a new transversality condition where the sum needs to be zero. The first term deals with the initial

conditions, the second with the final conditions, and the last term is for a variable end-time problem and disappears when a fixed-end time problem is examined. Note that if the initial state is independent of the final state, which is usually the assumed case, the terms can be separately set to zero. Also, Equations (C.9), (C.10), and (C.11) are identical to the case where no initial state constraint or control parameters are present.

REFERENCES

[1] J. Riehl, L. Mason, J. Gilland, J. Sovey, and H. Bloomfield, *Power and Propulsion Parameters for Nuclear Electric Vehicles.* NASA Lewis Research Center, Cleveland, Ohio, 1988.

[2] C. A. Kluever and B. L. Pierson, "Optimal earth-moon trajectories using nuclear electric propulsion," *Advances in the Astronautical Sciences*, vol. 90, pp. 1623–1638, 1996.

[3] J. Hermel, R. A. Meese, W. P. Rogers, R. O. Kushida, J. R. Beattie, and J. Hyman, "Modular, ion-propelled, orbit-transfer vehicles," *Journal of Spacecraft and Rockets*, vol. 25, no. 5, pp. 368–374, 1988.

[4] M. Guelman, "Earth-to-moon transfer with a limited power engine," *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 5, pp. 1133–1138, 1995.

[5] J. Riehl, L. Gefert, R. Myers, E. Pencil, J. McAdams, and D. Kusnierkiewicz, "Low power electric propulsion for minor body applications," in *31st AIAA/SAE/ASME/ASEE Joint Propulsion Conference*, AIAA-95-2812, (San Diego, CA), 1995.

[6] S. Borowski, *Nuclear Thermal Rocket Workshop Reference System.* NASA Lewis Research Center, Cleveland, Ohio, 1992.

[7] M. Cross, "The kennedy imperative," *New Scientist*, vol. 140, pp. 26–28, December 1993.

[8] L. Rothstein, "Now you see it, now you don't," *The Bulletin of the Atomic Scientists*, vol. 48, no. 3, p. 6, 1992.

[9] S. Borowski, "The rationale/benefits of nuclear thermal rocket propulsion for nasa's lunar space transportation system," in *27th AIAA/SAE/ASME/ASEE Joint Propulsion Conference*, AIAA-91-2052, (Sacramento, CA), 1991.

[10] S. Borowski, R. Corban, D. Culver, M. Bulman, and M. McIlwain, "A revolutionary lunar space transportation system architecture using extraterrestrial lox-augmented ntr propulsion," in *30th AIAA/SAE/ASME/ASEE Joint Propulsion Conference*, AIAA-94-3343, (Indianapolis, IN), 1994.

[11] G. Sachs and M. Dinkelmann, "Reduction of coolant fuel losses in hypersonic flight by optimal control," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 6, pp. 1278– 1284, 1996.

[12] M. Fukushima, "A successive quadratic programming algorithm with global and superlinear convergence properties," *Mathematical Programming*, vol. 35, pp. 253–264, 1986.

[13] R. Chamberlain, "Some examples of cycling in variable metric algorithms for constrained minimization," *Mathematical Programming*, vol. 16, pp. 378–383, 1979.

[14] R. Chamberlain, M. Powell, C. Lemarechal, and H. Pedersen, "The watchdog technique for forcing convergence in algorithms for constrained optimization," *Mathematical Programming Study*, vol. 16, pp. 1–17, 1982.

[15] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *Journal of Guidance, Control, and Dynamics*, vol. 16, pp. 59–68, January-February 1993.

[16] J. T. Betts and W. P. Huffman, "Application of sparse nonlinear programming to trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 15, pp. 198–206, January-February 1992.

[17] G. Parker and R. Rardin, *Discrete Optimization.* Boston: Academic Press, 1988.

[18] B. Borchers and J. Mitchell, "An improved branch and bound algorithm for mixed integer nonlinear programs," *Computers and Operations Research*, vol. 21, no. 4, pp. 359–367, 1994.

[19] K. Holmberg, "On the convergence of cross decomposition," *Mathematical Programming*, vol. 47, pp. 269–296, 1990.

[20] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications.* Topics in Chemical Engineering, New York, NY: Oxford University Press, 1995.

[21] I. Grossmann, "Minlp optimization strategies and algorithms for process synthesis," in *Foundations of computer aided process design* (J. Siirola, I. Grossman, and G. Stephanopoulos, eds.), p. 105, NASA Lewis Research Center, 1990.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing.* New York, NY: Cambridge University Press, second edition ed., 1992.

[23] B. Ferraris, *Building Libraries the Object Orientated Way.* New York: Addison Wesley, 1992.

[24] G. Dow, *Inside Power Plant for CW8.* Austin, Texas: Metrowerks, 1996.

[25] D. Mark, *Learn C++ on the Macintosh.* New York: Addison Wesley, 1993.

[26] B. L. Pierson and C. A. Kluever, "Three-stage approach to optimal low-thrust earth-moon trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1275–1282, 1994.

[27] M. Rivas and B. Pierson, "Dynamic boundary evaluation method for approximate optimal lunar trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 4, pp. 976–978, 1996.

[28] F. L. Lewis, *Optimal Control.* New York, NY: John Wiley & Sons, 1986.

[29] M. Rivas, "Minimum-fuel low-thrust lunar trajectories for a range of initial thrust-to-weight ratios," master of science thesis, Iowa State University, Ames, 1993.

[30] C. A. Kluever and B. L. Pierson, "Optimal low-thrust earth-moon transfers with a switching function structure," *Journal of the Astronautical Sciences*, vol. 42, no. 3, pp. 269–283, 1994.

[31] J. Lewins, *Nuclear Reactor Kinetics and Control.* New York, NY: Pergamon Press, 1978.

[32] G. C. Oates, *Aerothermodynamics of Gas Turbine and Rocket Propulsion.* AIAA Education Series, New York, NY: American Institute of Aeronautics and Astronautics, 1984.

[33] S. Glasstone and A. Sesonske, *Nuclear Reactor Engineering*, vol. 2. New York: Chapman and Hall, 4 ed., 1994.

[34] R. R. Mohler and C. Shen, *Optimal Control of Nuclear Reactors*, vol. 6 of *Nuclear Science and Technology.* New York: Academic Press, 1970.

[35] S. Borowski and S. Alexander, "'fast track' ntr system assessment for nasa's first lunar outpost mission," in *28th AIAA/SAE/ASME/ASEE Joint Propulsion Conference*, AIAA-92-3812, (Nashville, TN), 1992.

[36] C. Watson, *Nuclear Rocket Propulsion.* Gainesville, Florida: Florida Engineering and Industrial Experiment Station, 1964.

[37] A. Zimmerman, "Simulation of ntr start-up and cooldown," tech. rep., Analex Corporation, Cleveland, OH, 1992.

[38] M. Rivas, "Nuclear-thermal rocket thrust transient effects on minimum-fuel lunar trajectories," in *12th Symposium on Space and Nuclear Power and Propulsion*, (Albuquerque, NM), 1995.

[39] S. Glasstone and A. Sesonske, *Nuclear Reactor Engineering*, vol. 1. New York: Chapman and Hall, 4 ed., 1994.

[40] T. V. Roy, "Cross decomposition for mixed-integer programming," *Mathematical Programming*, vol. 25, p. 46, 1983.

[41] J. Benders, "Partitioning procedures for solving mixed-integer programming problems," *Numerical Mathematics*, vol. 4, p. 238, 1962.

[42] A. Geoffrion, "Generalized benders decomposition," *Journal of Optimal Theory and Control*, vol. 10, no. 4, p. 237, 1972.